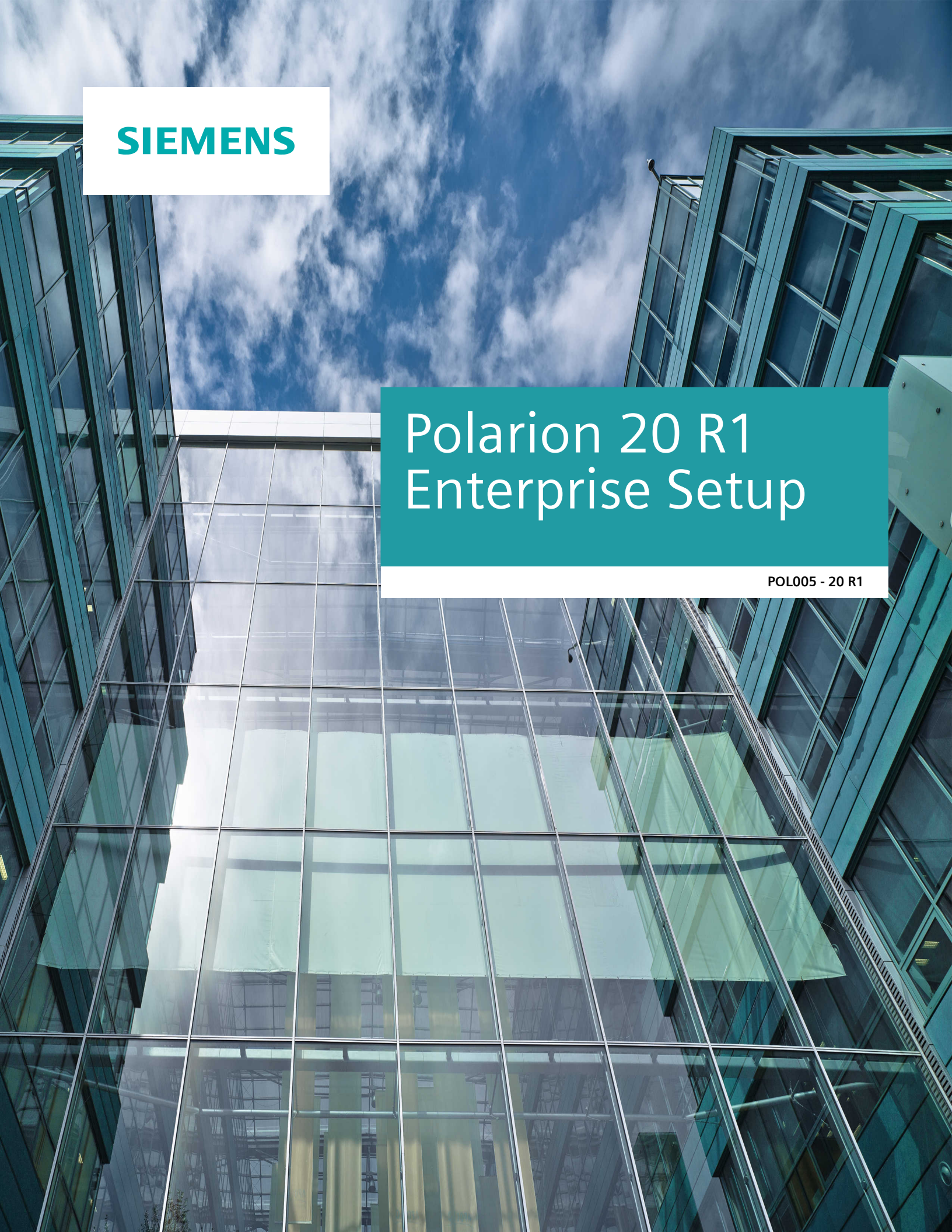


SIEMENS

Polarion 20 R1 Enterprise Setup

POL005 - 20 R1

Contents

1. Terminology

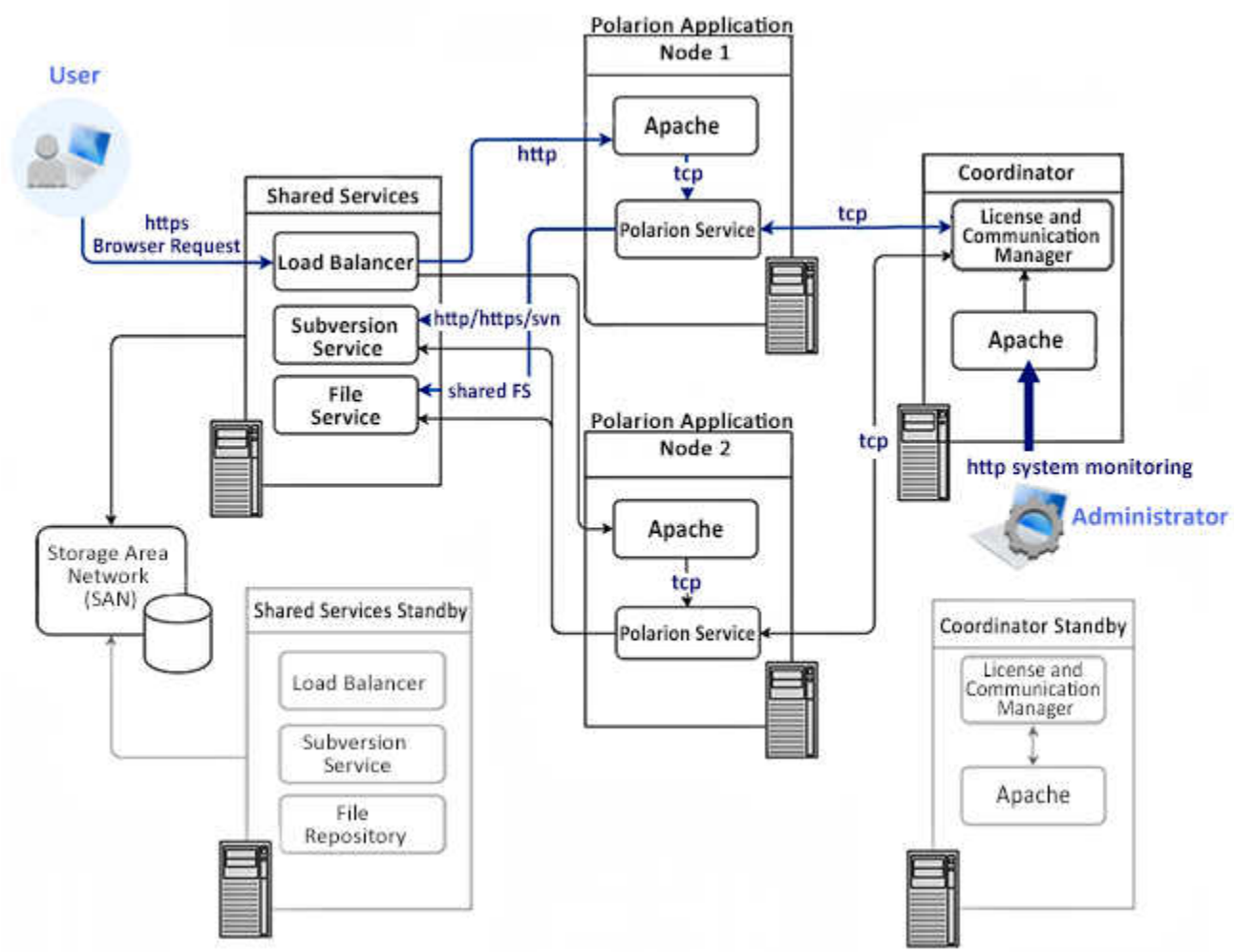
- **Polarion** - General reference to the **Polarion ALM** software/server/system.
- **Instance** - Polarion installation for a single node or a Polarion cluster for HA (high availability).
- **Stand-alone Instance** - An Instance, with its own repository on the same machine (i.e. not clustered), exposed to the user as one Polarion Server. This is also known as a single node instance.
- **Cluster** - A group of Polarion application nodes accessing a single instance of data. To the user, a polarion cluster acts as one logical Polarion Server representing a single instance of data.
- **Coordinator** - A specially configured Polarion installation that manages communication and licenses among **Instances**.
- **Shared Services** - A machine that hosts the Subversion repository, shared service folder (SSF) for shared data and load balancer (user entry point). There is one shared services per Cluster. The core glue for a Polarion cluster is the SSF that contains the common `polarion.properties` file.
- **Common Polarion Properties file** - A central Polarion systems property file (`polarion.properties`) stored in the SSF that contains most if not all of the core Polarion configuration for the cluster. The application nodes' local `polarion.properties` file should contain enough information to bootstrap the application node to read the common properties file. Each of these nodes will contain at least a directive like one of the following that defines `com.polarion.shared`.

```
########################################
#Shared folder between the machines that make up the cluster.
#default Linux: com.polarion.shared=opt/polarion/shared
#default Windows: com.polarion.shared=c:\Polarion\shared
```

- **Subversion Service** - Is a server service that provides access to a Subversion versioned file system. This can be bundled with the shared services or it can be on a stand-alone server.
- **HA (High Availability)** - A higher degree of online availability and connection resilience over single node single instance.
- **Polarion Application Node (PAN)** - A Polarion Application node represents the physical components that make up the logical Polarion Application stack, Apache, PostgreSQL RDBMS and Polarion Service.

2. Overview

The following figure shows one clustered setup with two clustered instances sharing one repository.



3. Details

A cluster setup requires one dedicated physical or virtual machine for each of the following: **Coordinator**, **Shared Services** and at least one or more Polarion Application Nodes called hereafter **Application Node**.

Coordinator

- Distributes tasks that need to be executed on a single **Application Node** in the **Cluster**.
- Serves as the Licensing Server for all **Application Nodes** connected to the **Coordinator**.
- Provides a single entry point to all logical Polarion servers that are connected to the same **Coordinator**.
- Reconfigures the Load Balancer if some of the instances are offline.

Shared services

- Provides the Load Balancer that forwards users to a specific **Application Node** in the **Cluster**.
- The entry point for a single **Cluster**.
- Provides a file repository shared by all **Application Nodes** in the **Cluster**.
- Serves the Subversion repository that contains the data for the clustered logical Polarion server.

Application Node 1, Application Node 2

- Machines running the Polarion service, connected in a **Cluster** and all configured to use the same shared services.
  (Each **Application Node** in the **Cluster** uses the same Subversion repository.)
- Every **Application Node** in the **Cluster** has its own Polarion data (indexes, object maps), and PostgreSQL database.
  (A shared database is not currently supported.)

4. Requirements

Server software requirements

Several virtual or physical machines are needed: one for the **Coordinator**, one for every **Application Node** (stand-alone or from a **Cluster**) and one **Shared Services** per **Cluster**.

The server software requirements are the same for all machines, as described in the *Windows* and *Linux* installation guides. The latest 2.4.x version is recommended.

Although the Coordinator machine does not really need Subversion, it is still recommended to use the standard Polarion Installer to install Polarion on it. It will install Subversion on the coordinator, and can just remain there.

IMPORTANT!

All machines (Coordinator, Nodes, and Stand-alone Instances) **must be running the same versions of Polarion and Java**.

Requirements for a Windows installation

A Polarion clustered setup in a Microsoft Windows environment requires the following:

- MS Active Directory.
- A DNS service is recommended, but you can also use static IP addresses. (The same subnet is expected.)
- Testing for a proper network configuration by 'pinging' between all hosts in a cluster.
- A domain user. (example, *yourDomain\polarion*)
  - For shared Services, CIFS/Samba requires a domain user, for example, *yourDomain\polarion*.
- A mail server or mail GW is required for sending e-mail notifications.
- For proper configuration, you will need to open ports as described in *Server hardware*.

Server hardware requirements

Overview

Generally the requirements for all server machines are similar to those described in the Polarion installation guides for *Linux* and *Windows*. The PDF and HTML versions are available in the Polarion section of **Siemens' Doc Center**.

All machines must be connected by a fast 1 Gbps low-latency (< 10 ms) intranet network. Enterprise-class installations should be using at least a 10 Gbps interface preferably channel bonded interfaces for fail-over and load balancing to network switches.

IMPORTANT!

Polarion is a transaction-oriented system and the performance of Polarion is predicated on having fast storage. Polarion does not recommend using pooled common storage for the Polarion RDBMS or data folder. It's OK for the operating system to be installed on storage from a common hypervisor storage pool, but for Polarion RDBMS and data folder this storage should be on SSD (Solid State Disk) and be connected to the operating system using either ISCSI, RDMs or something similar to these fast connection methods.

Coordinator server

Requirement	Description
CPU	2
RAM	2-4 GB
Disk space	50 GB
FQDN	e.g *coordinator.mycompany.com*
Access from clients (http(s))	The **Coordinator** provides signpost and server monitoring pages. Choose the port (usually 80 for http, 443 for https) and configure Apache and Polarion to use them. **Related Polarion property:** `base.url=http(s)://host:port` (FQDN or IP address)
Access from Instances	Communication between instances and the **Coordinator** (ZooKeeper) takes place on the TCP/IP port specified by the `com.polarion.zookeeper.port` property of the coordinator. It is

Requirement	Description
	2181 by default. This port on the Coordinator host must be accessible by all instances. **Related Polarion properties:** `com.polarion.zookeeper.port=port#` (On the Coordinator.) `com.polarion.zookeeper=host:port#` (On the Instances.)

Stand-alone Instance server

Requirement	Description
CPU	See the "**Example Hardware Configurations**" table.
RAM	See the "**Example Hardware Configurations**" table.
Disk space	See the "**Example Hardware Configurations**" table.
FQDN	e.g. *myserver1.mycompany.com*
Access from clients (http(s))	Choose the port (usually 80 for http and 443 for https), then configure Apache, and Polarion to use them. **Related Polarion property:** `base.url=http(s)://host:port` (Must be FQDN or IP address.)

Access to Subversion	The same as a simple installation. There should be http(s) access from clients (end users), and svn protocol access is recommended for fast local access by system users.
	Related Polarion properties:
	`repo=http(s)://host:port/repo`
	`repoSystem=[svn/file/http(s)]://host:port`

Clustered Application Node

In Polarion's clustered environments, customers are looking for high availability with the need to support a growing concurrent user count. Polarion recommends that you scale out linearly with **Polarion Application Nodes** rather than scale a single node up to support higher concurrency.

IMPORTANT!

Disk I/O will be critical as the number of concurrent users begins to increase. On the Polarion Application Node (PAN) there are two key disk I/O services. The Polarion PostgresSQL RDBMs and the Lucene Indexes. The Polarion core will also perform a lot of write data logging of its activities. In enterprise-class environments, it will be important to provide unique disk for both PostgreSQL RDBMs and the Lucene Indexes. These disks should both be SSD disks to reduce latency. The connected I/O channels should be using the fasted protocol, Polarion recommends the use of ISCSI, RDMs or something similar in your virtualization environment. If none of these technologies are available then a dedicated storage pool should be created for use by your Polarion cluster and performance monitoring of disk latency should be carefully managed.

Requirement	Description
CPU	See the "**Example Hardware Configurations**" table.
RAM	See the "**Example Hardware Configurations**" table.
Disk space	See the "**Example Hardware Configurations**" table.
Time synchronization	A system time that is synchronized with all other cluster instances.

Access from Load Balancer	Load balancer needs to be able to redirect the requests to the cluster instances using the http(s) or AJP port where Polarion is running.
	The following should be located within the Common Polarion Properties file `polarion.properties` located in the **SSF**.
	Related Polarion properties:
	`base.url=http(s)://host:port`
	(Must be FQDN or IP address of Load Balancer.)
	`com.polarion.loadBalancer.workerUrl=http/https://host`
	(Host is the host of the application node.)
	If using AJP:
	`com.polarion.loadBalancer.workerUrl=AJP://host`

Requirement	Description
	(Host is the host of the application node.)
Communication between cluster Application Nodes (PAN)	RPC communication between **Polarion Application Nodes**(PAN) takes place on the TCP/IP port specified by the `controlPort` property of the PAN. All PANs of the **Cluster** must be able to access the control ports of all other PANs in the cluster. **Related Polarion properties:** `controlPort=port#` `controlHostname=host` (Must be FQDN or IP address)
Active Load Balancing Session Replication	When **Active Load Balancing** (ALB) is enabled, the cluster nodes share the session data with the other nodes. It uses a standard TCP connection and port 4000. When the ALB is disabled, the channel is not open. **Related Polarion Property:** `com.siemens.polarion.cluster.nodeHostname`

Active Load Balancing
Membership Management
	When **Active Load Balancing** (ALB) is enabled, the cluster nodes (the underlying Tomcat server) must be aware of each other. To do this, ALB uses a UDP multicast connection. When the ALB is disabled, the channel is not open.
	Related Polarion Properties:
	`com.siemens.polarion.cluster.membershipChannel.address`
	`com.siemens.polarion.cluster.membershipChannel.port`
	`com.siemens.polarion.cluster.nodeHostname`
	IMPORTANT!
	This ALB communication channel requires that multicast networking is enabled and configured between the cluster nodes. Please check with your network administrators and make sure that your network configuration and network policy allows for the use of multicast in your environment.

Shared Services server

Requirement	Description
CPU	4 (8 for XL)
RAM	8GB (16GB for XL)
Disk space	100GB (But can grow a lot depending on data.)
FQDN	For example:*myserver2.mycompany.com*
Access from Clients to Load Balancer (http(s))	The entry point to the cluster is the Load Balancer. Choose the http(s) protocol, configure Apache and adjust the common polarion configuration file for the cluster. **Related Polarion properties:** `base.url=http(s)://host:port` (On Instances - Must be an FQDN or IP address.)

Access from Coordinator to Load Balancer manager (http(s))	The Coordinator communicates with the Load Balancer manager via http(s).
	Configure Load Balancer manager application location in Apache.
	Related Polarion properties:
	`com.polarion.loadBalancer=http(s)://host/balancer-manager`
	(On **Application Nodes**.)
	`com.polarion.cluster.#ClusterId#.loadBalancer.user=`
	(On **Coordinator**.)
	`com.polarion.cluster.#ClusterId#.loadBalancer.password=`
	(On **Coordinator**.)

Shared Services Folder (**SSF**)	This folder is also known as the **Shared Servcies Folder**(SSF) because it is shared onto each of the **Application Nodes**. (Linux paths are used. For Windows, use analogical paths.) Folder `/opt/polarion` of **Shared Services** must be mounted as `/opt/polarion/shared` on all cluster instances.

Requirement	Description

	Note:
	This folder sharing should be set up after the installation of Polarion.
	
	User "`polarion`" on all nodes must have read access for: `/opt/polarion/shared/**`, and write access for at least the following:
	
	`/opt/polarion/shared/data/svn/*`
	
	`/opt/polarion/shared/data/BIR/**`
	
	`/opt/polarion/shared/data/RR/**`
	
	`/opt/polarion/shared/data/workspace/**`
	
	Note:
	The Index is located on the nodes in a high availability setup and not in the shared folders.
	
	(/* means files inside the directory, /** means everything including the subdirectories recursively.)
	
	Files created by the user on behalf of which the polarion service is running (**polarion**) on any node in `/opt/polarion/shared` must be readable by the a user on behalf of which the Apache server is running on the shared services.
	
	Thumbnail Storage for attachment previews are found in:
	
	`/polarion/shared/data/workspace/previews-data/thumbnails`
	
	Related Polarion property:
	
	`com.polarion.shared=/shared/directory/path` (on instances)

Access to Subversion	From clients (end users) and each instance of the **Cluster**, the Subversion repository must be accessible. Either the http(s) or svn protocols can be used. (Svn is recommended for fast access by system users). The following property should be configured in the common Polarion properties file `polarion.properties` located on shared services, but it can also be defined in each of the **Application Nodes** if preferred. **Related Polarion properties:** `repo=http(s)://host:port/repo` (On Instances) `repoSystem=[svn/http(s)]://host:port` (On Instances)

Example hardware configurations for Application Nodes

Environment	S	M	L	XL
Operating system	64 - bit	64 - bit	64 - bit	64 - bit
CPU cores	4	8	16	16
OS GB RAM (Polarion Java memory)	16 (8)	32 (16)	64 (32)	128 (64)
Storage for Polarion	500GB+	1TB+ (SCSi or similar)	1TB+ (RAID 10, NAS, SAN)	1TB+ (RAID 10, NAS, SAN)
# of Polarion Projects	< 300	< 500	< 750	< 1000
# Concurrent, logged-on users (on 1 instance)	< 30	< 60	< 100	< 150

Make sure that there is enough RAM available to the OS for file-caching. If the SVN is hosted on a different machine, more memory could be allocated for the Polarion process. Depending on the number of active Polarion projects and their configuration for per work item data tracking, the OS disk cache can require anywhere from 5 GB to 20+ GB of physical RAM for performance.

License requirements

If you host Polarion on your company's infrastructure, you must provide all physical hardware and/or virtual machines needed for the setup you want to implement, see *Installation use cases*, and obtain a license for the instances you will run.

Every node in a **Cluster** or server in a Multiple Stand-alone instances setup counts towards the **multiple instances** limit set in the license. Please contact the **Polarion ALM team** for assistance with any licensing questions.

Supported browsers and versions

Polarion ALM is web-based software. For the best experience, always use a supported browser. Polarion may work with other browsers, but only those listed here have been tested and certified as supported. Polarion displays a warning message to users who log in using an unsupported browser.

Browser	Version(s)	Notes
Google Chrome	Latest	
Mozilla Firefox	68 or newer	Version 74 for users hoping to tap into the very latest features and improvements. Organizations or individuals looking for longer-term support with less frequent browser updates should use Version 68 **ESR**
Microsoft Edge	Latest	
Microsoft Internet Explorer	11	Please be sure to review the additional notes section, below, for important notes about this browser.

Additional notes for Microsoft Internet Explorer

- If running Internet Explorer on Windows Server 2012 (for example, for testing):
  - If prompted by the browser, you will need to add **about:blank** as a trusted server.
  - You may need to add the local Polarion server and any remote Polarion servers as trusted servers.
  - You need to enable JavaScript for the Polarion server.

  > Note:
  >
  > The above points do not apply when accessing Polarion running on Windows Server 2012 from other clients.

- Users of Internet Explorer should specify their Polarion server in the Local Intranet security zone while ensuring that Compatibility View is not used.
- There can be performance issues on client computers running Internet Explorer due to browser memory requirements. Occasional restarts of the browser are recommended.

5. Installation use cases

Overview

This chapter describes simple use cases for *Cluster* and *Multiple stand-alone instances* setups. See *migrate a pre-2014 "multi-instance" setup* to learn how to migrate an older version of Polarion to the new configuration released with version 2014.

> Note:
>
> The **multi-instance** setup with local instances configured with a pre-2014 of Polarion still works with Polarion 2014 and newer release without any changes in the configuration. However, it is no longer possible to create new local instances.

If you want to configure a clustered instance from any of your local instances, then you need to migrate the whole setup to the new *Multiple stand-alone instances* setup, where instances always run on a dedicated machine.

Common terms

- **[INSTALL]** - The root directory of your current installation. This would typically be **C:\Polarion** on Windows or **/opt/polarion** on Linux.
- **[APACHE]** - Apache configuration directory. On Linux it should be **/etc/httpd/conf.d/** and on Windows **C:\Polarion\bundled\apache\conf\extra**.

Setting up a cluster from new installations

Options and prerequisites

Administrators can setup up either one of the following:

- A Cluster of installations (below)
- *Multiple stand-alone instances*

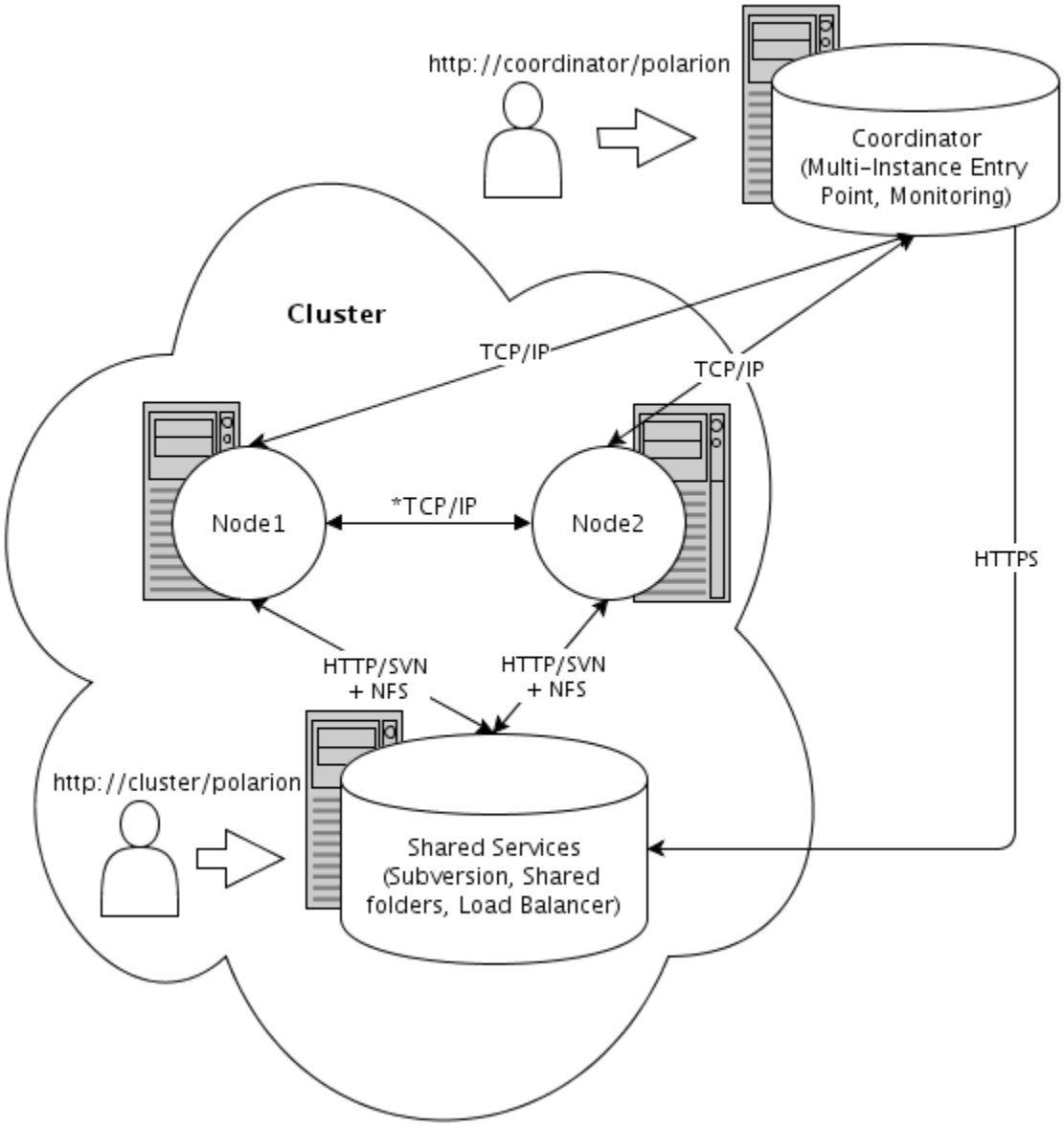
To set up a simple Cluster with two Nodes from new/clean Polarion installations:

(These machines must be running the same version of Polarion.) We recommend performing clean installs of stand-alone Polarion instances on all of the servers that make up the desired **Cluster**. Test each node independently to make sure it is working using a temporary license file. Conversion from a stand-alone instance to a cluster node is made simple with a change of configuration files.

Prerequisites: 4 machines (virtual or physical:)

1. **Coordinator** *(http://coordinator.yourdomain.com)*
2. **Node1***(http://node1.yourdomain.com)*
3. **Node2** *(http://node2.yourdomain.com)*
4. **Shared Services** *(http://cluster.yourdomain.com)*

Deployment diagram:



***** (When **Active Load Balancing** is enabled, Multicast traffic is also used.)

Start by installing the same version of Polarion on the following machines: coordinator, Node1, Node2 and Shared Services.

> Note:
>
> Different third-party software is required on individual machines:

- **On the Nodes**: Java, Apache HTTP Server, and PostgreSQL
- **On the Coordinator**: Java, and Apache HTTP Server
- **On the Shared Services**: Apache HTTP Server, and Subversion

The easiest way is to use the standard installation procedure to install all dependencies and eventually uninstall the software that is not needed if you need to save space on storage. All third-party software, except Java, is bundled in Windows distributions. It is already present on most Linux installations. Refer to the *Windows* or *Linux* installation guides for complete installation instructions. The PDF and HTML versions are available in the Polarion section of **Siemens' Doc Center**.

Instantiation of a local Subversion repository must only be done on the Shared Services machine since it is the only repository that will actually be used.

> Caution:
>
> Polarion should not be started immediately after installation, as further changes in configuration are required in order to set up the **Cluster**.

Because the coordinator serves as a license hub for all the nodes and instances connected to it, you do not need to activate any licenses on the nodes.

Once Polarion is successfully installed on each node and are specific for each machine, continue on with *Configuring the Cluster's coordinator*.

Configuring the cluster's coordinator

1. Stop Polarion.
2. Make a backup of the original **polarion.properties** file.
3. Replace **polarion.properties** using the coordinator template file provided in **[INSTALL]/polarion/ install folder: polarion.properties.template.coordinator**.
4. Make the following changes in the template-derived properties file, following the comments within the template:
   - Specify `base.url` appropriately for the machine. Must be FQDN or IP address.
   - Set the same value for `ajp13-port` as in the original **polarion.properties** file.
   - Set the same value for `controlPort` as in the original **polarion.properties** file.
   - Specify `controlHostname` appropriately for the machine.

5. *(Optional)* Uncomment the two properties about the load balancer credentials if the Apache load balancer is protected using basic authentication according to **Step 3** in the *Configuring the cluster's shared services section*. (User name and password).

   - The default setup uses the same credentials as the svn repository.

6. *(Optional)* Change the *ZooKeeper* port if the default port specified is not appropriate or blocked by a firewall policy.

7. *(Optional)* To disable the unused SVN repositories on the nodes, remove the **polarionSVN.conf** file from the Apache configuration directory and restart Apache.

   - The Apache configuration directory on Linux should be: `/etc/httpd/conf.d/`

   - The Apache configuration directory on Windows should be: `C:\Polarion\bundled\apache\conf\extra`

8. *(Windows only)* Make sure that the Polarion service is started with the credentials of a domain user created for Polarion. Use the same user for all Polarion installations.

9. Start Polarion.

Below is the configured **polarion.properties** file for the coordinator from the steps above:

```
com.polarion.application=polarion.coordinator
base.url=http://coordinator.yourdomain.com
TomcatService.ajp13-port=8889

# Control port and host name for shutdown requests
controlPort=8887
controlHostname=coordinator.yourdomain.com

# Credentials used to connect to the load balancer, if authentication is
enabled
# Replace #ClusterId# with the id of the cluster.
#com.polarion.cluster.#ClusterId#.loadBalancer.user=
#com.polarion.cluster.#ClusterId#.loadBalancer.password=

# Port to connect to zookeeper
com.polarion.zookeeper.port=2181
```

License deployment on coordinator

Polarion 2015 and later: Activate the Polarion license using the **Polarion Activation** window on the Coordinator. Accessing *http://coordinator.yourdomain.com/polarion* or the logon screen of any Node or

Instance will redirect you automatically to the **Polarion Activation** window. For more information see **Activation Help**.

Polarion 2014 and earlier: Make sure that the correct license file is placed in the license folder prior to starting the server:

- **On Linux: /opt/polarion/polarion/license/**
- **On Windows: C:\Polarion\polarion\license**

A Cluster's license is activated in the same way as a single instance (described in the Polarion Installation Guide documentation). The activation application runs on the Coordinator machine and instructs the user how to activate on-line or off-line. Users accessing any entry point and the login screens of individual nodes and instances are redirected to the activation page on Coordinator until the activation is complete. Nodes and instances can start even if Polarion is not activated, but users cannot log in.

Configuring the cluster's shared services

1. Stop Polarion server.

2. Uninstall the Polarion service.

   - On Linux, run this script: `/opt/polarion/bin/uninstall_polarion_service.sh`

   - On Windows, run: `C:\Polarion\polarion\service.bat -uninstall` (Requires administrator privileges.)

3. Configure the Load Balancer in Apache using the example template file provided in the **[INSTALL]/polarion/install** folder: **loadbalancer.conf.apache24.template**. Copy it to the `[APACHE]` directory and rename it to **loadbalancer.conf**.

   - Basic authentication is configured in the template and you need to check the correct location for the **AuthUserFile**.

4. (Windows only) Make sure that **loadbalancer.conf** is included in **httpd.conf**:

```
# Polarion
Include conf/extra/loadbalancer.conf
Include conf/extra/polarion*.conf
```

5. Comment out or remove the following lines from **polarion.conf**:

```
ProxyPass /polarion ajp://127.0.0.1:8889/polarion timeout=600
ProxyPassReverse /polarion ajp://127.0.0.1:8889/polarion
```

6. Make changes in the **loadbalancer.conf** file, following the comments provided:

   - Change path for **passwd** appropriately for this machine.
     On Linux it will be `/opt/polarion/data/svn/passwd`.
     On Windows it will be `C:\Polarion\data\svn\passwd`.

   - Adjust **BalancerMembers** to point to the address of each node in the **loadbalancer.conf** file.

   - Adjust **ProxyPassReverse** settings to point to the address of each node in the **loadbalancer.conf** file:

```
<Location "/polarion">
ProxyPass balancer://polarion_cluster/polarion timeout=900
```

```
ProxyPassReverse http://example-node1/polarion
ProxyPassReverse http://example-node2/polarion
</Location>
```

- Make sure that the **ProxySet** directive contains a timeout parameter.
  (For example, **timeout=600**.)
  If the parameter is not mentioned in **loadbalancer.conf**, append it to the end of a line.

- *(Optional)* Add the **keepalive=on** directive to all of the **BalancerMember** rows when you have a firewall between your Apache httpd and backend server. (Because the firewall tends to drop inactive connections.) This will tell the operating system to send KEEP_ALIVE messages on inactive connections to prevent the firewall from dropping the connections.

- *(Optional)* Uncomment logging directives if you want enable logging for the load balancer.

7. Restart Apache. ("`service httpd restart`" on Centos.)

8. Set up the **shared folder**:

   - On Linux machines, we recommend NFSv4 protocol for sharing.

   - On Windows machines, you can use CIFS/Samba share for sharing. It must be shared for the same domain user that is used for running all polarion installations in the cluster. The user needs all permissions for the share.

   - Data sharing for different Operating systems and protocols is covered in *Configuring shared data*.

9. Make backup of the original **polarion.properties** file on this machine.

10. Modify **polarion.properties**:

    - The original properties from the clean installation must be preserved. These properties will be shared between nodes in the cluster, so everything that is common to nodes should be there.

    - Add the `com.polarion.zookeeper=coordinator.yourdomain.com:2181` property.

    - Add the `com.polarion.clusterId=cluster1` property.

    - Add the `com.polarion.clusterLabel=Main Cluster` property.

    - Add the `com.polarion.clusterDescription=Description of Main Cluster` property.

- Add the `com.polarion.loadBalancer=http://cluster.yourdomain.com/balancer-manager` property.
- Modify the `svn.access.file=$[com.polarion.shared]/data/svn/access` property.
- Modify the `svn.passwd.file=$[com.polarion.shared]/data/svn/passwd` property.
- Modify the `polarion.build.default.deploy.repository.url= file://$[com.polarion.shared]/data/shared-maven-repo` property
- Comment out the `repoSystem` property.
- Comment out the `com.polarion.platform.internalPG` property.

> Note:
>
> The `http://cluster.yourdomain.com/balancer-manager` URL must point to the Apache Load Balancer Manager URL. The domain is machine-specific and will be used as the entry point for this Cluster.

> Note:
>
> The `com.polarion.platform.internalPG` property must be present in all nodes of the **polarion.properties** file.

Below is the configured `polarion.properties` for **Cluster** - **Shared Services** (this properties file will be used by each node in the **Cluster**). This is file is known as the **Common Polarion Properties File** (CPPF).

> Note:
>
> For production installations, it is recommended that you configure the `repoSystem` property to use the **svn** protocol to communicate with the Subversion Service. See Optimize Subversion in Polarion Help for details on how to set up the svnserve service.

```
# Newly added properties to original file
com.polarion.zookeeper=coordinator.yourdomain.com:2181
com.polarion.clusterId=cluster1
com.polarion.clusterLabel=Main Cluster
com.polarion.clusterDescription=Description of Main Cluster
com.polarion.loadBalancer=http://cluster.yourdomain.com/balancer-manager

# Modified properties
#repoSystem=...
# Subversion Service (svnserve) for system user access on Shared
Services Server
```

```
#repoSystem=svn://polarion.mycompany.com/

#
# Define shared SSF location of Subversion access & password files
svn.access.file=$[com.polarion.shared]/data/svn/access
svn.passwd.file=$[com.polarion.shared]/data/svn/passwd
polarion.build.default.deploy.repository.url=file://$
[com.polarion.shared]/data/shared-maven-repo

#
# Subversion Service (Apache HTTPD) for end user access on Shared
Services Server
repo=http/https://polarion.mycompany.com/repo
```

Configuring the cluster's Application Nodes

Configuration steps

IMPORTANT!

The following steps must be performed **for each Node in the Cluster**.

1. Stop the Polarion server.
2. Make a backup of the original **polarion.properties** file.
3. Replace **polarion.properties** using the example template file provided for nodes in the **[INSTALL]/polarion/install** folder: **polarion.properties.template.node**
4. Make sure that the shared folder is mounted on this machine on the recommended path:
   - Shared folder on **Linux** should be in `/opt/polarion/shared` .
   - Shared folder on **Windows** is accessed directly as `\\<shared_services_host>\Polarion`.
5. Make changes in the template file following the comments provided:
   - Set `com.polarion.shared` to point to the mounted shared folder:
     On **Linux** it should be `/opt/polarion/shared`.
     On **Windows** it should be `\\\\<shared_services_host>\\Polarion`.
   - Set the same value for `ajp13-port` as in the original **polarion.properties** file.
   - Set the same value for `controlPort` as in the original **polarion.properties** file.

- Set the `controlHostname` value to *node1.yourdomain.com* or *node2.yourdomain.com*. (Depending on which node you are configuring.)

- Set the value for `com.polarion.loadBalancer.workerUrl` to the specific node in cluster so that the load balancer knows the URL of the node.

- Set a value in the `calc.base.url` property to a specific node in the Cluster. It must point to the specific node, otherwise calculation will fail. It is the same as `workerUrl`. (For example, `calc.base.url=http://node1.yourdomain.com`)

- Add the `com.polarion.platform.internalPG` property with a value from the **Cluster's** shared services where it is commented out. If the PostgreSQL configuration is identical on each **Application Node** a best practice would then be to have this property defined in the **Common Polarion Properties File** (`polarion.properties`) on the **SSF** and not defined on each of the **Application Nodes**.

6. *(Optional)* To disable the unused SVN repositories on the nodes, remove the **polarionSVN.conf** file from the Apache configuration directory and restart Apache.

   - The Apache configuration directory on **Linux** should be `/etc/httpd/conf.d/`.

   - The Apache configuration directory on **Windows** should be `C:\Polarion\bundled\apache\conf\extra`

7. *(Windows only)* Make sure that the Polarion service is started using credentials of a domain user created for Polarion. The same user should be used for all Polarion **Application Nodes**. This is required so they can share a common file system.

Below is the configured **polarion.properties** file for Node1. (It will be the same for second or third Nodes, except that the URLs must be changed accordingly.)

```
# Shared folder between the machines that make up the cluster
# default Linux: com.polarion.shared=/opt/polarion/shared
#default Windows: com.polarion.shared=\\\\\\\\\\\\<shared_services_host>\
\\\\\Polarion
com.polarion.shared=/opt/polarion/shared
com.polarion.nodeId=node1
TomcatService.ajp13-port=8889

#Url of node in load balancer
com.polarion.loadBalancer.workerUrl=http://node1.yourdomain.com

# Control port and host name for shutdown requests
controlPort=8887
controlHostname=node1.yourdomain.com
```

```
#Node-specific url of the node.
#It is used in calculations to access Polarion via web services
#calc.base.url=http://example-node
calc.base.url= http://node1.yourdomain.com

#Postgres database connection (Optionally can be defined in the common
Polarion properties file
#instead of here).
com.polarion.platform.internalPG=polarion:passwordForDatabase@localhost:5
433
```

IMPORTANT!

Even if you plan on using the *Resource Traceability* feature, add the following property to the shared **polarion.properties** file:

```
com.siemens.polarion.rt.startRtServer=false
```

Once the cluster is setup, but before setting up a *Standalone Resource Traceability Server*, remove the property from the **polarion.properties** file.

You have now configured an entire cluster for a clean installation of Polarion.

Your cluster is accessible on: *http://cluster.yourdomain.com*

Server Monitoring is accessible on: *http://coordinator.yourdomain.com/polarion/monitoring*

The Apache Load Balancer Manager is accessible on: *http://cluster.yourdomain.com/balancer-manager*

Synchronizing time on cluster nodes

> Caution:
>
> Time must be synchronized on each node in the cluster on the OS level by a system administrator. Ideally this should be an automated sync via NTP. If the time is not synchronized, users will see different times on each node, scheduled jobs may appear to start off schedule and the **Monitor** will incorrectly order jobs by time.

Configuring the cluster's activation application

Beginning with version 2015, Polarion includes an activation application that makes it possible to install or update a license, while the Polarion server is running, without the need to copy the license file manually to the target machine. Access to this application is *NOT* initially protected by a user name and password. For production use, it is highly recommended to secure access to this application directly in the Apache configuration. It is only necessary to perform this configuration on the coordinator server.

In version 2015 installations there is a template Apache configuration file in the Polarion installation folder:

/polarion/polarion/install/polarion.activation.conf.template

To ensure that a user name and password is requested when accessing the activation application

(`/polarion/activate/online and /polarion/activate/offline`), copy this file to the Apache configuration folder.

On Linux usually `/etc/httpd/conf.d/`.

On Windows, usually `C:\Polarion\bundled\apache\conf\extra\`.

After copying the file, rename it to remove the **.template** extension. Then open the file in any text editor and modify it according to the instruction comments provided.

The template configuration is prepared for both user file authentication (like Polarion uses for Subversion by default, with user passwords data in a file) and for authentication against an LDAP server.

Multiple stand-alone instances setup

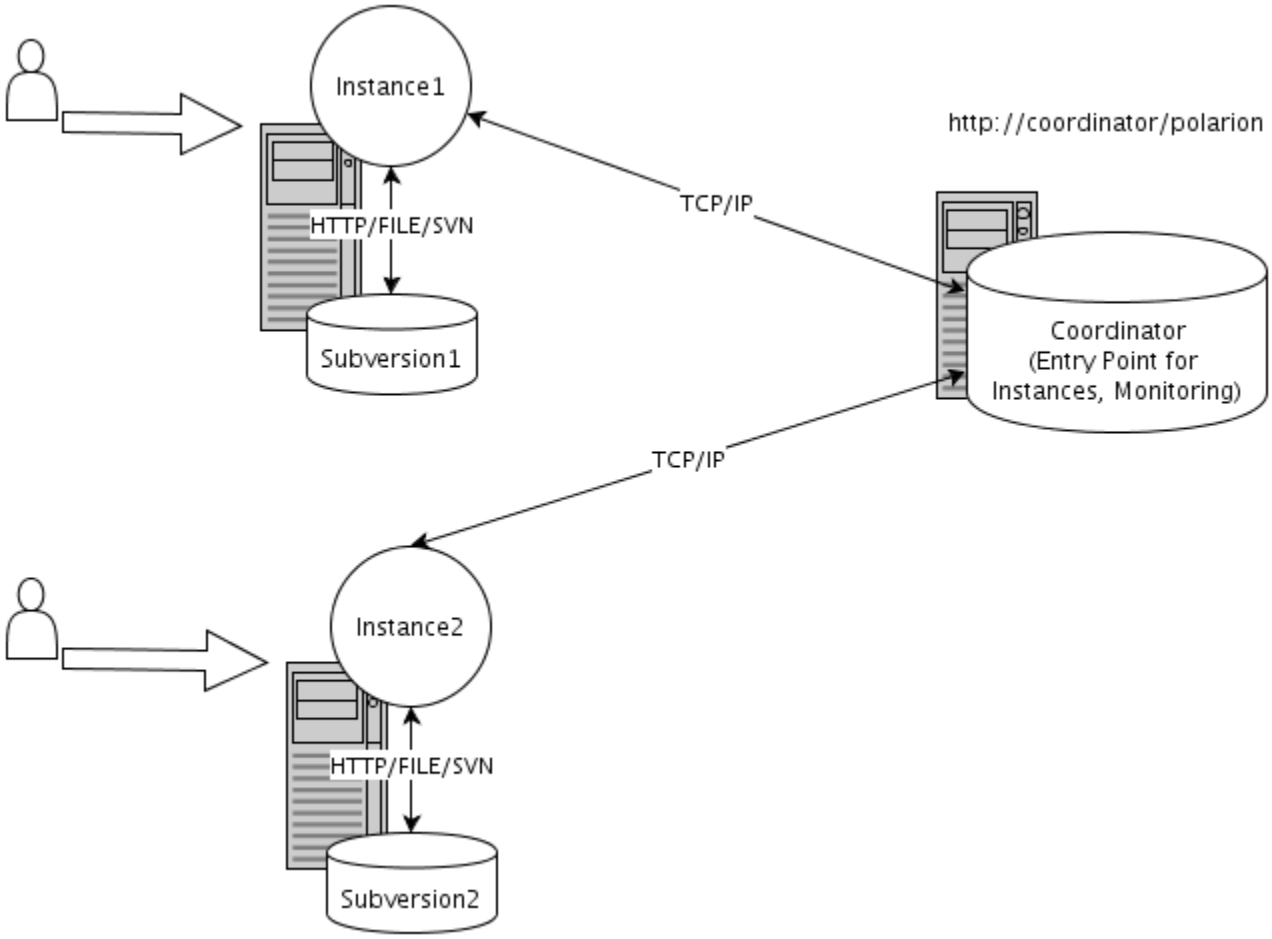
Using the coordinator for license management

You can set up the multiple stand-alone instances configuration using the coordinator for license management.

Three machines (virtual or psychical) are required for this setup:

1. Coordinator (*http://coordinator.yourdomain.com*)
2. Stand-alone Instance1 (*http://instance1.yourdomain.com*)
3. Stand-alone Instance2 (*http://instance2.yourdomain.com*)

Deployment diagram for multiple stand-alone instances:



Start by installing the **same version** of Polarion on each of the 3 machines: Coordinator, Instance1, and Instance2. The following different third-party software is required on the individual machines.

- On the instances - Java, Apache HTTP Server, Subversion, and PostgreSQL
- On the coordinator - Java, and Apache HTTP Server

The easiest way is to use the standard installation procedure to install all dependencies then uninstall the software that is not needed if you need to save storage space. All third-party software, except Java, is bundled in Polarion distributions for Windows. They are already present on most Linux installations. Refer to the Polarion installation guide for your preferred operating system (*Windows Installation* and *Linux Installation*) for complete installation instructions . The PDF and HTML versions are available in the Polarion section of **Siemens' Doc Center**.

The next sections assume that Polarion is successfully installed using the standard installation and running on each machine.

There are specific next steps that need to be performed on each machine.

Configuring the coordinator for multiple stand-alone instances setup

Configuration of the coordinator is exactly same as the cluster setup described in *Configuring the cluster's coordinator*. Proceed to configure the coordinator for this setup as described there.

The next section on Instances configuration will refer to this coordinator machine *(http://coordinator.yourdomain.com)* and assumes that the coordinator is configured and running.

Configuring Instance 1

On the machine hosting the Polarion installation for Instance 1:

1. Stop Polarion.
2. Make a backup of the original **polarion.properties** file.
3. Modify **polarion.properties** by adding the following but be sure that all properties in the original file are preserved:
   - Add the `com.polarion.zookeeper=coordinator.yourdomain.com:2181` property.
   - Add the `com.polarion.clusterId=Cluster1` property.
   - Add the `com.polarion.nodeId=Instance1` property.
   - Add the `com.polarion.clusterLabel=First Server` property.
   - Add the `com.polarion.clusterDescription=Description of first Server` property.
4. Start Polarion.

Below is the configured **polarion.properties** file for Instance1:

```
# Newly added properties to original file

com.polarion.zookeeper=coordinator.yourdomain.com:2181

# com.polarion.clusterId - is it identificator on coordinator
```

```
# (instance displays as independent cluster)

com.polarion.clusterId=Cluster1

com.polarion.nodeId=Instance1

com.polarion.clusterLabel=First Server

com.polarion.clusterDescription=Description of first Server

# List of properties from original file

repo=…

repoSystem=…

etc..
```

Configuring Instance 2

On the machine hosting the Polarion installation for Instance2:

1. Stop Polarion.
2. Make a backup of the original **polarion.properties** file.
3. Modify **polarion.properties** by adding the following but be sure that all properties in the original file are preserved:
   - Add the `com.polarion.zookeeper=coordinator.yourdomain.com:2181` property.
   - Add the `com.polarion.clusterId=Cluster2` property.
   - Add the `com.polarion.nodeId=Instance2` property.
   - Add the `com.polarion.clusterLabel=Second Server` property.
   - Add the `com.polarion.clusterDescription=Description of second Server` property.
4. Start Polarion.

Below is the configured **polarion.properties** file for instance2:

```
# Newly added properties to original file
com.polarion.zookeeper=coordinator.yourdomain.com:2181
# com.polarion.clusterId - is it identificator on coordinator
#  (instance displays as independent cluster)
com.polarion.clusterId=Cluster2
com.polarion.nodeId=Instance2
com.polarion.clusterLabel=Second Server
com.polarion.clusterDescription=Description of second Server

# List of properties from original file
repo=...
repoSystem=...
etc..
```

The configuration is quite similar to the cluster setup. The difference is that there is no load balancer or shared services. Each **Instance** is autonomous, a stand-alone Polarion installation with its own SVN repository. **Stand-alone Instances** have nothing to do with other **Instances** in a Multiple Stand-alone Instances setup. However, users can easily switch between the instances by accessing the entry point on the coordinator. You can also monitor the availability of each Instance using server monitoring.

> Note:
>
> The Polarion user interface (UI) and end-user documentation use the term **server** when referring to what we term **instance** for administrators. For example, the UI provides end-users the possibility to **Change Server**. In administration terms; to work on a different **Instance**.

Access URLs for multiple stand-alone instances

Entry Point (all Instances): *http://coordinator.yourdomain.com/polarion*

Server Monitoring: *http://coordinator.yourdomain.com/polarion/monitoring*

Instance1 direct access: *http://instance1.yourdomain.com/polarion*

Instance2 direct access: *http://instance2.yourdomain.com/polarion*

Migrating From a Pre-2014 multi-instance installation

Differences between the new and old multiple stand-alone instances" setups

Several versions prior to version 2014 supported a topography of multiple Polarion instances that was termed a "Multi-Instance" setup. Instance clustering was not supported. Although existing customer installations of this setup have still been usable with versions 2014 - 2016, the setup was documented in this guide as deprecated in favor of the many improvements delivered beginning with version 2014.

Warning:

Beginning with version 17, the pre-2014 multi-instance setup with local instances is no longer supported. If you are using this type of setup, in order to use version 17 (and subsequent versions), you must migrate your system to a multiple stand-alone instances setup as described here. (Review *Multiple stand-alone instances setup* to familiarize yourself with this setup). Customers with a current support and maintenance package may consult technical support for assistance with this migration.

The new Multiple Stand-alone Instances setup differs from old Multi-Instance setup in the following ways:

- The master is replaced by the coordinator, which manages a license for all instances.
- Local instances are not compatible with the new **Multiple stand-alone instances** setup. If you have local Instances configured and wish to update to 2017 (or later) multiple stand-alone instances, these local instances must be moved to separate machines and then configured later as part of a multiple stand-alone instances setup. (See *Moving local instances for the 2014 multi-Instance setup* for more information.)
- Each remote instance will become a non-clustered Instance connected to the coordinator.
- The coordinator does not start up the instances. They must be started individually.

In order to do the migration, you need to update Polarion on the old master and remote instances to the same version. Then you need to modify the configuration files so that they reflect the new configuration properties.

For example, a pre-2014 setup with one Master application and two Remote Instances will become a multiple stand-alone instances setup with one coordinator and two non-clustered instances. Each instance hosts a stand-alone installation of Polarion, complete with third-party software and a repository.

Configuring the coordinator

To replace the pre-2014 multi-Instance setup you need to configure the coordinator. The coordinator still runs on the machine where the master and local instances ran.

Follow the steps described in *Configuring the cluster's coordinator*, and also use the information from the **_controller.properties** file if needed. For example, `controlPort` and `controlHostname` can be taken from the **_controller.properties** file.

From this point on, it is assumed that you have the coordinator configured, running and accessible through following URL: *http://coordinator.yourdomain.com/polarion*.

Migrating a remote instance to a non-clustered stand-alone instance

1. Stop the instance and update it to the latest Polarion version (2017 or later).

2. Make a backup of the original **polarion.properties** file.

3. Add the following properties to the **polarion.properties** file but make sure all its original properties are preserved:
   Add the `com.polarion.zookeeper=coordinator.yourdomain.com:2181` property.
   Add the `com.polarion.clusterId=OldRemoteInstanceId` property.
   Add the `com.polarion.nodeId=OldRemoteInstanceId-node1` property.
   Add the `com.polarion.clusterLabel=Old Remote Instance Label` property.
   Add the `com.polarion.clusterDescription=Description of the old remote instance` property.

4. If you have any properties configured in **instanceid.properties**, they should be moved into **polarion.properties**, otherwise they will be ignored.

5. Start Polarion.

Below is an example of a **polarion.properties** file for a migrated remote instance. (The instance ID is *instance1*.)

```
# Newly added properties to original file
com.polarion.zookeeper=coordinator.yourdomain.com:2181
com.polarion.clusterId=instance1
com.polarion.nodeId=node1
com.polarion.clusterLabel=Remote instance - Instance1
com.polarion.clusterDescription=Description of the remote instance

# List of properties from original file
repo=…
repoSystem=…
etc..
```

Checking the migration

To check that the migration was successful, go to *http://coordinator.yourdomain.com/polarion* and connect to the instances.

- Entry point URL: *http://coordinator.yourdomain.com/polarion*
- Server monitoring URL: *http://coordinator.yourdomain.com/polarion/monitoring*

Each Instance can be still directly accessed through its URL: For example, *http://instance1.yourdomain.com/polarion*

The old configuration files for the pre-2014 Multi-Instance setup from **[polarion_installation]/configuration/multi-instance/*** will become obsolete.

Moving local instances for the multiple stand-alone instances setup

Moving a Local Instance refers to moving the existing repository and the configuration files to a new Polarion installation.

This step is only required if some of the Instances are configured as Cluster. If no Cluster is needed, the Local Instances will still work as they did before in the old Multi-instance setup with the same configuration.

Linux paths:

- The **polarion.properties** file: **opt/polarion/etc**.
- The repository folder: **/opt/polarion/data/svn** or **/opt/polarion/data/multi-instance/instanceId/svn**.

Windows paths:

- The **polarion.properties** file: **C:\Polarion\polarion\configuration**.
- The repository folder: **C:\Polarion\data\svn** or **C:\Polarion\data\multi-instance\instanceId\svn**

To move a local instance to a new machine:

1. Install Polarion on the new machine. Make sure it can be started correctly, then stop it and keep it stopped for the next steps.
2. In the new installation location, make a backup of the repository folder. This copy will subsequently be referred to as **svn_backup**.
3. In the new installation location: make a backup of the **polarion.properties** file. This copy will subsequently be referred to as **polarion.properties_backup**.
4. Perform an SVN dump on the existing repository.
   (`$ svnadmin dump /var/svn/repos > full.dump`. See **svnbook.red-bean.com** for details.)
5. Then Perform an SVNadmin load on the new repository.
   (`$ svnadmin load /var/svn/restored < repos-backup`. See **svnbook.red-bean.com** for details.)
6. Copy the **access** and **passwd** files to the new repository.

7. Copy the **polarion.properties** file from the old instance to the same location on the new machine. (See path references above.)

8. Start Polarion. You should have all the data from the old instance.

9. After a successful startup, you can delete **svn_backup** and **polarion.properties_backup** files.

At this point you have a clean installation of the latest Polarion version holding the data and configuration of the old instance. You can configure this instance as part of a multi-instance setup following the steps described in *Setting up a cluster from new installations*.

Updating a multiple stand-alone instance or cluster setup

When updating either setup, you can use the **update distribution** to update the machines in the setup (see steps below). To limit the downtime for the update of a **Cluster**, it is recommended that you start with updating of one of the **Application Nodes** of the **Cluster** up-front to minimize the downtime for the duration of the coordinator update.

IMPORTANT!

Logging on to an **Instance** or **Application Node** that has a different version of Polarion installed than the coordinator that it is connected to, is not supported. However, it is possible to run the reindex procedure on such a machine.

Update steps for a cluster:

1. Stop the Polarion service on one of the application nodes of the **Cluster**.

2. Check that there are no running **PhantomJS** and **Variants** server processes on the **Application Node** and if so, kill them.

3. Install the update on the target **Application Node** of the **Cluster**.

4. Start the Polarion service on the **Application Node** of the **Cluster** in reindex mode and wait for the reindex to finish.

5. Stop the Polarion service on all **Application Nodes** of a **Cluster**, including the one that has already been updated. (Stop the Polarion service on the **Coordinator** once it has been done on all the **Application Nodes**.)

6. Check that there are no running **PhantomJS** and **Variants** server processes on any of the instances and if there are, then kill them.

7. Install the update on the **Coordinator** machine and start Polarion in reindex mode.

8. Start the Polarion **Application Node** of the **Cluster** that was updated in **step 3**. Once it starts, your users can log on to your **Cluster**.

9. Install the update and start each of the remaining **Application Nodes** of the clustered instance in reindex mode.

> Warning:
>
> Ensure that the update of the SVN repository is only done once, either by updating one of the nodes up-front, or by updating the Cluster nodes in sequence.

> Warning:
>
> Updating the shared services machine is only required to update the bundled Apache and Subversion for Windows environments.

> Note:
>
> To update multiple stand-alone instances, all instances and the coordinator must be shut down and updated at once.

> Note:
>
> Running the reindex procedure and the DB History Creator job in parallel on multiple nodes puts a substantial load on the shared services machine and will prolong the reindex and DB History Creator run.

6. Configure shared data

Shared data configuration steps for both Windows and Linux

The following details how to configure shared data on Linux and Windows machines and the differences between the two.

Prerequisites (4 machines, all on the same domain):

1. Coordinator (*coordinator.yourdomain.com*)
2. Node1 (*node1.yourdomain.com*)
3. Node2 (*node2.yourdomain.com*)
4. Shared Services (*cluster.yourdomain.com*)

The shared folder has same structure as standard Polarion installation folder, so it is possible to use a Polarion installer to create it

1. Install Polarion.
2. Uninstall the Polarion service and delete the folders that are not needed. Only two folders in the Polarion installation are needed for shared data:
   - **Linux**: **/opt/polarion/etc** and **/opt/polarion/data**.
   - **Windows**: **C:/Polarion/polarion/configuration** and **C:/Polarion/data**.

> Note:
>
> The deletion of the other, unnecessary, folders is optional. You can also leave the installation folder as it is after installation.
>
> The root of the Shared Services is the **polarion** folder.

Linux configuration

Share the folders among the nodes **NFSv4** protocol. Other protocols (such as SSHFS or NFSv3) have known problems, so they must not be used.

NFS configuration

The following describes an example on how to set up folder sharing using the NFS protocol.

1. Connect to the Shared Services machine (*http://cluster.yourdomain.com*).

2. Edit the **/etc/exports** file and add the following lines:
   /opt/polarion node1(rw,sync,no_root_squash,no_subtree_check)
   /opt/polarion node2(rw,sync,no_root_squash,no_subtree_check)

3. On the Node machines create a **/opt/polarion/shared** folder.

4. On the Node machines add following line to the **/etc/fstab** file.
   cluster.yourdomain.com:/opt/polarion /opt/polarion/shared nfs defaults 0 0

5. Run the following commands on the Cluster, Node1 and Node2 machines. (Not the Coordinator.) :
   (Check `/etc/centos-release/` to identify your version.)
   For Centos 6.x Distributions:
   ```
   # /etc/init.d/portmapper start
   ```
   Portmapped was exchanged for **rpc bind for RH6 distributions**. Run `yum -y install nfs-utils` to install it.
   ```
   # /etc/init.d/rpcbind start
   # /etc/init.d/nfs start
   ```
   For Centos 7.x Distributions:
   ```
   # systemctl start nfs-config.service
   # systemctl start nfs-server.service
   ```

6. On the shared services machine run the following command:
   ```
   # exportfs -a
   ```

7. And on the node machines mount the shared directory with the command:
   ```
   # mount -v cluster.yourdomain.com:/opt/polarion /opt/polarion/shared/
   ```

8. Check that the **shared** folder appears on each node in the **/opt/polarion** folder, and make sure that the **polarion.properties** file on each node points to this location: **/opt/polarion/shared**.

9. Make sure that each node has **rw** permissions for the **/opt/polarion/shared** folder, and all nodes create folders and files with the same permissions.

Windows configuration

We recommend using standard Windows sharing on this platform.

CIFS / Samba share configuration

You need to configure sharing on the shared services machine, and all Node machines, starting with the shared services machine.

Shared services machine

A simple example on how to create the shared folder using CIFS/Samba.

1. Connect to the shared services machine (*http://cluster.yourdomain.com*).
2. Open File Explorer.
3. Right-click on the **C:/Polarion** folder.
4. Select **Properties**.
5. Select the **Sharing** tab .
6. Click **Share...** .
7. Set shared user as the same domain user for all polarion installations in the cluster. The user needs full permissions for the folder.
8. After you have configured the sharing options, click on **Share**, and then **Done**.

Node machines

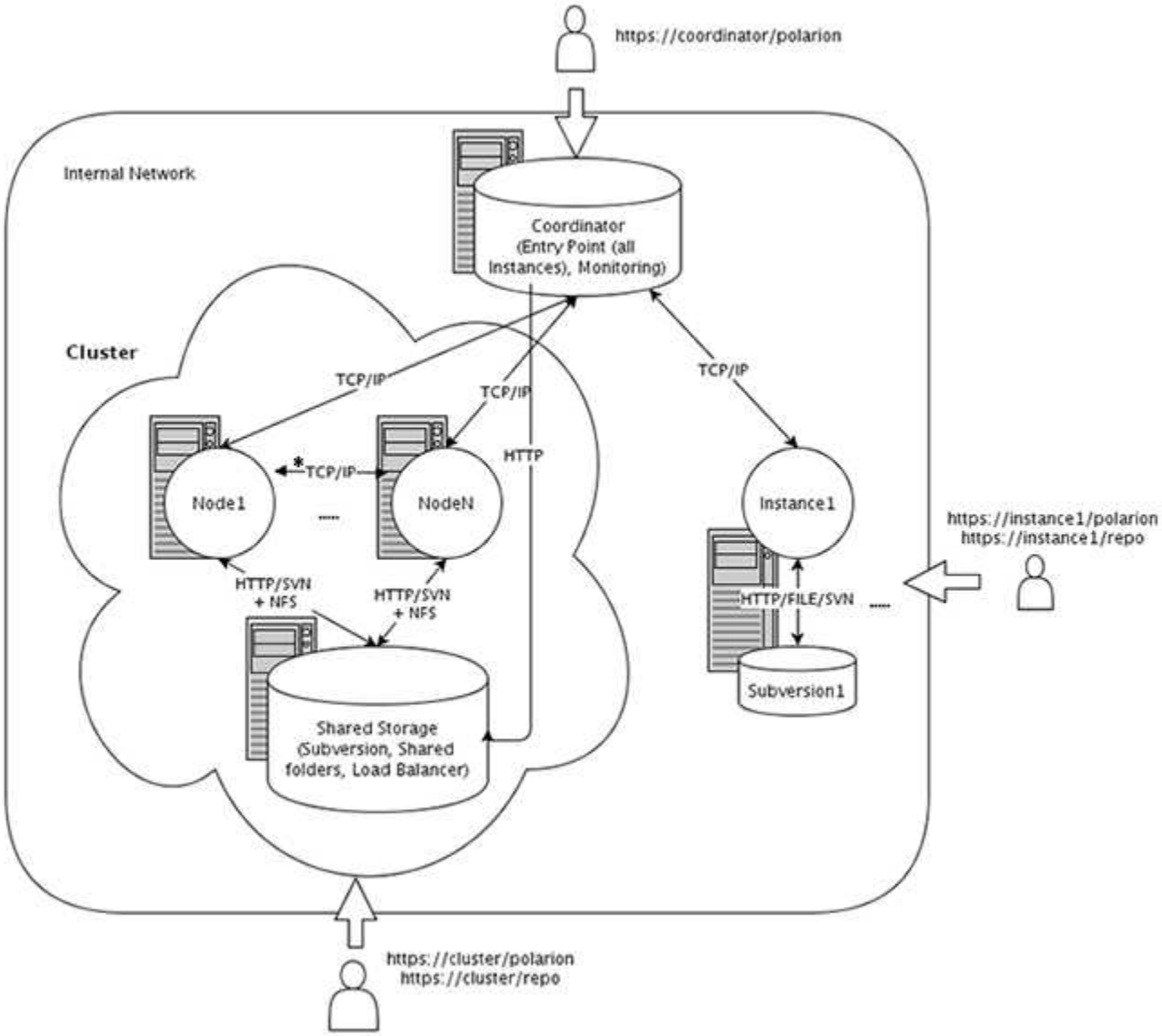
1. Connect to each Node (*http://node1.yourdomain.com* and *http://node2.yourdomain.com*)
2. Open File Explorer.
3. In the left panel, right-click on **Computer**.
4. Map a network drive. Using the credentials of the polarion domain user.
   - The folder should be: *cluster.yourdomain.com**polarion*
5. Edit the **polarion.properties** file accordingly and specify the path to the shared folder.

- The **com.polarion.shared** property must point to this mapped drive.

7. Security options

Recommended setup

The recommended setup is to use encrypted communication between the outside world and the internal network with servers of the Multiple stand-alone instances setup (as shown in the figure below). This is also optimal from a performance point of view. Communication inside the local network can optionally be encrypted as well. (Except for the folders shared using NFS, **Active Load Balancing** session replication and membership management.) See *Advanced security options* for more details.



***** (When **Active Load Balancing** is enabled, Multicast traffic is also used.)

Note:

HTTPS access should be set up in an Apache server. See some hints and references on how to do this in the **Administrator's Guide → Configuring SSL Support** section in Polarion's Help.

Use of anti-virus (AV) software

When running on server OS platforms that run instances of Polarion server, anti-virus (AV) software intercepts I/O requests to Polarion data structures in order to scan the data looking for virus signatures. This inserts latency (time delay) into all underlying file system read and write operations, which has the potential to directly impact Polarion Server performance, potentially in the magnitude of hundreds of percent.

Also, the AV software may detect false positives in Polarion data structures, which can result in data corruption when the AV software attempts to either re-write the data, or worse, quarantines data files. Any of the various methods used by AV software to deal with false positives could potentially result in data corruption.

Best practice guidance is to use caution when implementing AV software products on a server that hosts Polarion ALM, as it can impose performance and stability issues that may lead to poor response times or data corruption. Where feasible, not running AV software on well protected and/or network isolated server platforms assures that there will be no impact on Polarion server operation. Where AV software must be running, then at minimum it is strongly suggested to exclude the underlying file system supporting a Polarion server's Subversion repository and PostgreSQL database from real-time checking and/or dynamic scanning.

Appropriate security hygiene restricting outside access to the underlying file system supporting a Polarion server's Subversion repository is recommended practice that effectively moderates the need for AV data protections. Assuring that attachments to Polarion data structures (work Items, etc.) are only allowed from sources that *are* subject to AV data protections is an equally prudent security measure that effectively moderates need for AV data protections for Subversion data.

Warning:

Siemens disclaims liability for corruption of any Polarion data structure that is caused by running AV software on platforms supporting a Polarion server.

Recommended security options

Service	Security Settings
Entry point (Coordinator)	The entry point, where users can select the Polarion server, should be configured for HTTPS access in Apache so that end users will access, for example, *https://coordinator.mycompany.com/polarion*. Additional steps: Remember to update the `base.url` in the **polarion.properties** file.
Server monitoring (Coordinator)	The same as above for the server monitoring page, for example, *https://coordinator.mycompany.com/polarion/monitoring*. This will be usually done by the same configuration as the entry point.
Stand-alone instance Polarion	Standard HTTPS setup like is done for a simple stand-alone installation, so that the instance can be accessed as, for example, *https://instance1.mycompany.com/polarion*. If the Subversion repository is accessed by end users, it should be configured for HTTPS access as well. Additional steps: Remember to update the `base.url` in the **polarion.properties** file.
Clustered Polarion Instance	Standard HTTPS setup in Apache for the load balancer so that the clustered **Application Nodes** can be accessed as, for example, *https://nodeX.mycompany.com/polarion*. Where nodeX is the hostname for the Application Node. If the subversion repository is accessed by end users, it should be configured for HTTPS access as well. **Additional steps:** 1. Set the `wikiProtocolSchema=https` Polarion property in the shared cluster properties file (**/opt/polarion/etc/polarion.properties**) on the shared services machine.) 2. Remember to update the `base.url` in the shared cluster properties.

Advanced security options

If desired, the internal network communication among the servers comprising the multiple stand-alone instances setup can be encrypted as well.

Service	Security Settings
Load balancing	Load balancing communication between the load balancer and the workers (clustered **Application Nodes**) can be done via HTTPS if desired. HTTPS access must be set up on the **Coordinator** and all cluster **Application Nodes**, (as it is for a simple installation), and then configure the load balancer to use the HTTPS worker URLs. You can use the same wildcard certificate on all servers. **Additional steps:** 1. It is necessary to switch on the SSL proxy engine using **SSLProxyEngine on** in the Apache configuration. (`httpd/conf/httpd.conf` on Centos distributions.) 2. The `wikiProtocolSchema=https` property must be set in the shared cluster properties file (**/opt/polarion/etc/polarion.properties**), on the shared services machine. 3. Remember to update the `base.url` in shared cluster properties. Note: By default, Apache does not verify the certificate of the workers. To switch it on, set `SSLProxyVerify=require` property and you might also need to set the `SSLProxyCACertificatePath` property or other directives. For more details, see Apache's **mod_ssl** documentation: **http://httpd.apache.org/docs/2.4/mod/mod_ssl.html**.
Load balancer management	By default, the **Coordinator** manages the load balancer. For example, it switches off the worker if a Polarion **Application Node** disconnects from the **Cluster**. This management is done using the HTTP/HTTPS URL provided by the `com.polarion.loadBalancer` shared cluster property. The load balancer manager is a web application provided by Apache, and it can be configured for HTTPS access on shared services. **Additional steps:**

Service	Security Settings
	1. Remember to update the `com.polarion.loadBalancer` Polarion property the in shared cluster properties or often called the **common Polarion properties** file. 2. It might be necessary to install a trusted certificate authority to the Java trust store, especially if a self-signed certificate is used.
Subversion repository	For the best performance, clustered **Application Nodes** should access the shared repository by the system user using the SVN protocol (**repoSystem=svn://...**). To do so, svnserve running on the shared services machine, must be open to remote access. This communication is not encrypted. To enhance security you may want to consider establishing a secure channel using, for example, **Stunnel** . The idea is that instead of... **repoSystem=svn://SHARED_SERVICES_HOST/opt/polarion/data/svn/repo** Use... **repoSystem=svn://localhost/opt/polarion/data/svn/repo** On the cluster node and connect **localhost:3690** to **SHARED_SERVICES_HOST:3690** by a secured channel. **Another option:** To enhance security you can create a private network using VLAN that links the **Coordinator**, each of the **Application Nodes** and the Subversion server together. This is a pretty easy and straight forward recommended best practice approach.
Cluster coordination	Instances and nodes in the multiple stand-alone instances setup communicate with the coordinator machine. This communication is not encrypted. It can be secured using, for example, **Stunnel** in a very similar way to that described above. Cluster instances also communicate directly with other cluster instances using a TCP socket on the **controPort**. This communication is encrypted internally. **Another option:** You can create a private network using VLAN that links the **Coordinator**, each of the Polarion **Application Nodes** or Polarion Services with the **Coordinator**.

Authentication for server monitoring

After the initial installation of a cluster, the **Server Monitoring** page is available on the coordinator. On this page, administrators can view and access all the configured nodes (servers), and manage the load balancer. The access URL for this page is as follows: *http://coordinator.yourdomain.com/polarion/monitoring*.

The page does not require any authentication. However, authentication is recommended and you can configure basic authentication via the standard way in Apache, using one of the following directives:

(Assuming that the password file is either **/opt/polarion/etc/passwd** for Linux or **C:/Polarion/data/svn/passwd** for Windows.)

Apache 2.4 and newer:

```
<Location /polarion/monitoring>
  Require all denied
  AuthType Basic
  AuthName "Monitoring"
  AuthUserFile "C:/Polarion/data/svn/passwd"
  Require valid-user
</Location>
```

8. Using Resource Traceability in a cluster

Before setting up Resource Traceability

> Warning:
>
> To ensure that a cluster setup installs correctly, the `com.siemens.polarion.rt.startRtServer=false` property was added to the shared **polarion.property** file.

This property should be removed before setting up a Resource Traceability server.

Standalone Resource Traceability server

To configure a cluster or standalone Resource Traceability installation connected to a Polarion cluster:

> Note:
>
> This configuration is recommended to ensure the high-availability of the Resource Traceability server.
>
> If the Resource Traceability node goes down, it can be quickly restarted without having to restart the Polarion application itself.

Adjust database

1. Shutdown Polarion and PostgreSQL.
2. Go to the **[POLARION_DATA]/postgres-data** folder.
3. For Windows or Linux installations, open the **postgreql.conf** file and comment out the following properties and uncomment the entry for the same property exactly below them:
   - `max_connections`
   - `shared_buffers`
   - `work_mem`
   - `maintenance_work_mem`
   - `fsync`
   - `synchronous_commit`

- `full_page_writes`
- `wal_buffers`
- `checkpoint_segments`
- `effective_cache_size`
- `max_locks_per_transaction`

4. Restart PostgreSQL.

To create a database on a new location - to have it on shared storage - please contact **Polarion support**.

> Note:
>
> To connect a Resource Traceability Server to an external database, the following should be used:

```
com.siemens.polarion.rt.db.jdbcUrl=jdbc:postgresql://
<databaseLocation>:5433/polarion

com.siemens.polarion.rt.db.username=<username> (e.g polarion)

com.siemens.polarion.rt.db.password=<password>
```

Adjust the resource traceability server's polarion.properties file

When connecting the Resource Traceability server to a Polarion Cluster.

1. Mount the shared storage to the Resource Traceability node. (Required to share the **polarion.properties** file.)

2. Make a copy of your **polarion.properties** file for Resource Traceability.

3. After making the copy, replace its content with the content below and adjust the properties if needed:

```
com.siemens.polarion.rt.polarionUrl=http://polarion-cluster
com.polarion.application=polarion.rt
#Shared folder between the machines that make up the cluster
#default Linux: com.polarion.shared=/opt/polarion/shared
#default Windows: com.polarion.shared=\\\\<shared_services_host>\
\Polarion
com.polarion.shared=/opt/polarion/shared
TomcatService.ajp13-port=8889
```

```
base.url=http://rt-hostname
# Control port and host name for shutdown requests
controlPort=8887
controlHostname=rt-hostname
com.polarion.platform.internalPG=polarion:polarion@localhost:5433
```

> Note:
>
> The *com.siemens.polarion.rt.polarion* URL should point to the cluster address that goes through the load balancer.

HTTPS setup is done like any other Polarion instance. Certificates must also be imported into the truststores of both the Polarion and Resource Traceability servers.

Adjust the virtual memory settings

1. Adjust the Virtual Memory properties so that they fall into the **-Xms500m -Xmx2g** range. These values will vary depending on the number of external repositories, their size and scheduling.

   - For Windows: In the **[POLARION_HOME]/polarion.ini** file.
   - For Linux: In the **[POLARION_HOME]/etc/config.sh** file.

2. Restart Polarion.

Adjust the Polarion server

Adjust the Polarion server to work with the Standalone Resource Traceability server.

When connecting a Polarion cluster to a Standalone Resource Traceability Server, add the following properties to each node:

```
com.siemens.polarion.rt.startRtServer=false

com.siemens.polarion.rt.url=http://rt-hostname
```

> Note:
>
> `com.siemens.polarion.rt.url` should point to the `base.url` of the standalone Resource Traceability server. (For both cluster and single installations.)

HTTPS setup is done like for any other polarion instance. Additionally, import the certificates into the truststores of both Polarion and Resource Traceability server.

Embedded Resource Traceability server in cluster nodes

To ensure a high-availability setup, use the *Standalone resource traceability* setup.

> Warning:
>
> To ensure that a cluster setup installs correctly, the `com.siemens.polarion.rt.startRtServer=false` property was added to the shared **polarion.property** file. It should be removed before setting up a Resource Traceability server.

To correctly configure a Resource Traceability cluster, setup **Reader** and **Writer** nodes.

- **Reader Node:** Can only return links that are stored in the Resource Traceability database for a specified **Work Item**. There is no limit to the number of Reader nodes located in the cluster.
- **Writer Node:** Enables configuration updates, collects information from the repositories and stores data, files, configurations and links in the database. Only a single Writer node is allowed in the cluster.

> Note:
>
> Writer node settings can be left as is because a Polarion instance starts the Resource Traceability sever by default as a Writer instance.

Configure reader nodes

Customize the following Properties to use a different PostgreSQL instance for storing links:

A database on a different node acts like a separate PostgreSQL instance and the properties below should also be provided on the node or instance pointing to the database.

com.siemens.polarion.rt.db.jdbcUrl=jdbc:postgresql://someurl

`(e.g. node2):5433/polarion` where the URL points to a different server.

By default `com.polarion.platform.internalPG` is used to fetch database properties.

`com.siemens.polarion.rt.db.jdbcUrl`

`com.siemens.polarion.rt.db.username`

`com.siemens.polarion.rt.db.password`

All Reader nodes should be configured to send different write requests to the Writer node.

They should also all be marked as Reader nodes by setting the `com.siemens.polarion.rt.dataCollectingEnabled=false` property to **false**.

`com.siemens.polarion.rt.writerNodeUrl=` should be linked to the Writer node's base URL.

Define the same database properties in the following properties for every Reader node. They should be linked to the Database that is used by the Writer node. This enables the Polarion located on a Reader node to send a request to fetch Work Item links for its local RT Server instance along will all other requests, for example, configuration changes to the Writer node.

```
com.siemens.polarion.rt.db.jdbcUrl

com.siemens.polarion.rt.db.username

com.siemens.polarion.rt.db.password
```

9. Active Load Balancer

Prior to 18.2, cluster nodes handled user sessions independently. If a node was shut down or failed, a user would be redirected to another node, but would be required to log on again and risked losing some of the data that they were working on. Now, with session replication enabled, if a node becomes unavailable, another can seamlessly take over an existing user session without the need to log back in or losing unsaved data.

```
com.siemens.polarion.cluster.activeLoadBalancingEnabled
```

Enables session replication in a cluster environment. When Polarion is set up in a cluster, the Active load balancing can either:

- Be set to `false` - The default setting for existing installations and the only option prior to Polarion 18.2.
  Keep sessions on the nodes separate so that moving a user from one node (load balancing, failover) to another, triggers a login pop-up if **Remember me** is not checked on the new node.

- Be set to `true` - Enables Active load balancing.
  Session data is replicated between cluster nodes so that moving a user from one node to another is handled seamlessly, without the need to log in again.

> Note:
>
> When using node session replication, the nodes must be able to reach each other via a TCP/UDP connection. See more details below.

```
com.siemens.polarion.cluster.membershipChannel.address
```

And...

```
com.siemens.polarion.cluster.membershipChannel.port
```

These two parameters define the cluster membership communication channel multicast address and the port that distinguishes between multiple Polarion clusters on a single network. It uses the UDP protocol. The default values for the parameters are `228.0.0.4` for the address parameter and `45564` for the port parameter. (The same as the default Tomcat parameters.) If you are running a single Polarion cluster, the values can be omitted. You should also set these parameters if they and other, non Polarion Tomcat servers, are in a cluster configuration on the same network.

Note:

You don't need to set or modify both parameters. In most cases, only setting one, is enough.

(But your network configuration must allow multicast addressing within the clustered networks in order to enable the cluster node management.)

IMPORTANT!

Cluster membership management communication requires that multicast networking be enabled and configured between the cluster nodes. Please check with your network administrators and make sure that your network configuration and network policy allows for the use of multicast in your environment.

```
com.siemens.polarion.cluster.nodeHostname
```

This parameter represents a node local bind host name for the Tomcat inter-node TCP communication channel that's responsible for session replication. It's a node specific value and should be set to the correct network interface host name or IP address.

This is an optional parameter with an **auto** default value that translates into value retrieved via the `java.net.InetAddress.getLocalHost().getHostAddress()` java call. Because the host has multiple network interfaces, including a loop back interface, the Java implementation takes the first address from the unordered(!) list of local addresses.

If session replication doesn't work with the default configuration, check the following INFO level debug output:

```
TomcatAppServer cluster local bind host name: 'some.hostname.company.com'
```

If the host name is incorrect, override the bind host name with the above parameter with the correct value.

```
com.siemens.polarion.cluster.excludedSessionAttributes
```

This parameter represents a list of Java classes that are declared as not serializable. (They don't implement **java.io.Serializable**.) This means that they're NOT replicable across cluster nodes. It's a list of fully qualified java class names separated by commas and without spaces(!) .

Administrators should list all non-serializable classes that they use there for:

- Custom Rich Page page scripts,
- Their own extension modules.

The replication of a session that contains non-serializable attributes can cause issues with the replication process and active load balancing. (They can be identified in the Polarion log files by searching for **forjava.io.NotSerializableException**.)

Warning:

Adding an attribute to the list will make the session replicable, but won't include the added attribute. This might affect the Rich Page page scripts or the used extension module session data and the way that the replicated node(s) function.

Warning:

To ensure that session replication works with all data, adjust your page scripts and/or extension modules so that they only store serializable attributes into the session data.

```
com.siemens.polarion.cluster.stickyCookieMaxAge
```

This parameter defines the user session stickiness cookie timeout and expiration. It's used by the Active Load Balancing feature to enable the distribution and balancing of users across the cluster nodes. The parameter is only used when Polarion is configured in a cluster and when the Active Load Balancing is switched on.

Tip:

Set the parameter when the Active Load Balancing is switched on, otherwise users will be stuck on a single node "forever". (As long as the cluster node is alive and the user is logged on.)

The value is in seconds and the default value is 300 (5 minutes). Administrators can experiment with the value to find an optimal value that suits their setup. (A recommended range is between 120 and 600 seconds.)

Configure sticky cookie

```
com.siemens.polarion.cluster.stickyCookieName
```

Set the name of the cookie that defines how "sticky" the Apache session is and request routing in a cluster environment.

Note:

This parameter is only used when Polarion is configured in a cluster and when the Active load balancing is enabled. Otherwise it's ignored.

If the Active Load Balancer enabled and configured to use sticky sessions, all of a user's interactions will happen with the same physical server, even in a multi-server setup.

IMPORTANT!

- The value must the same as the cookie name configured in the Apache load balancer configuration.

(The default **ROUTEID** value is compatible with the default Apache load balancer configuration templates.)

- If the name of the sticky cookie in the `polarion-cluster.conf` and `polarion.properties` file are different, then two cookies will be generated (one from the Polarion server and one from the Apache cluster configuration), and they will not work.

- If the Active Load Balancer is not active, the sticky cookie from Polarion will not be used.

10. Notes and troubleshooting

Notes

- **Web Services**: With the 17.1 release, Polarion's web service client supports load balancing in a cluster. (External applications that connect to Polarion via web services should use the load balancer's URL as the entry point to Polarion.)

- **Extensions** are not shared among the nodes in a cluster. Each node has its own independent extensions folder (e.g. **/opt/polarion/polarion/extensions**). However, while an extension can be installed on specific node(s) in a cluster, there are only a few, unique extensions that don't need to be installed on all nodes.

  > Warning:
  >
  > Any extension that contributes to Polarion's UI (stores data in user sessions) **MUST** be installed on all nodes.

- **Scheduled jobs** should be reviewed for a cluster, and convenient node selectors (i.e. the node attribute of **<job>** elements) should be specified depending on the nature of the job. The following default jobs should have **node="*"** specified: **Index Checker**, **Suspend DB History Creator**, **Resume DB History Creator**.

- **Diagnostics**: Polarion comes with a self-diagnostic utility **Polarion Diagnostic Tool** (PDT), which can run comprehensive diagnostic tests and communicate the results to Polarion's technical support team. PDT checks if Polarion is running in a cluster and gathers configurations from shared folders. The utility is located in the **diagtool** folder under the root of any installed Polarion instance, that also contains documentation for its use.

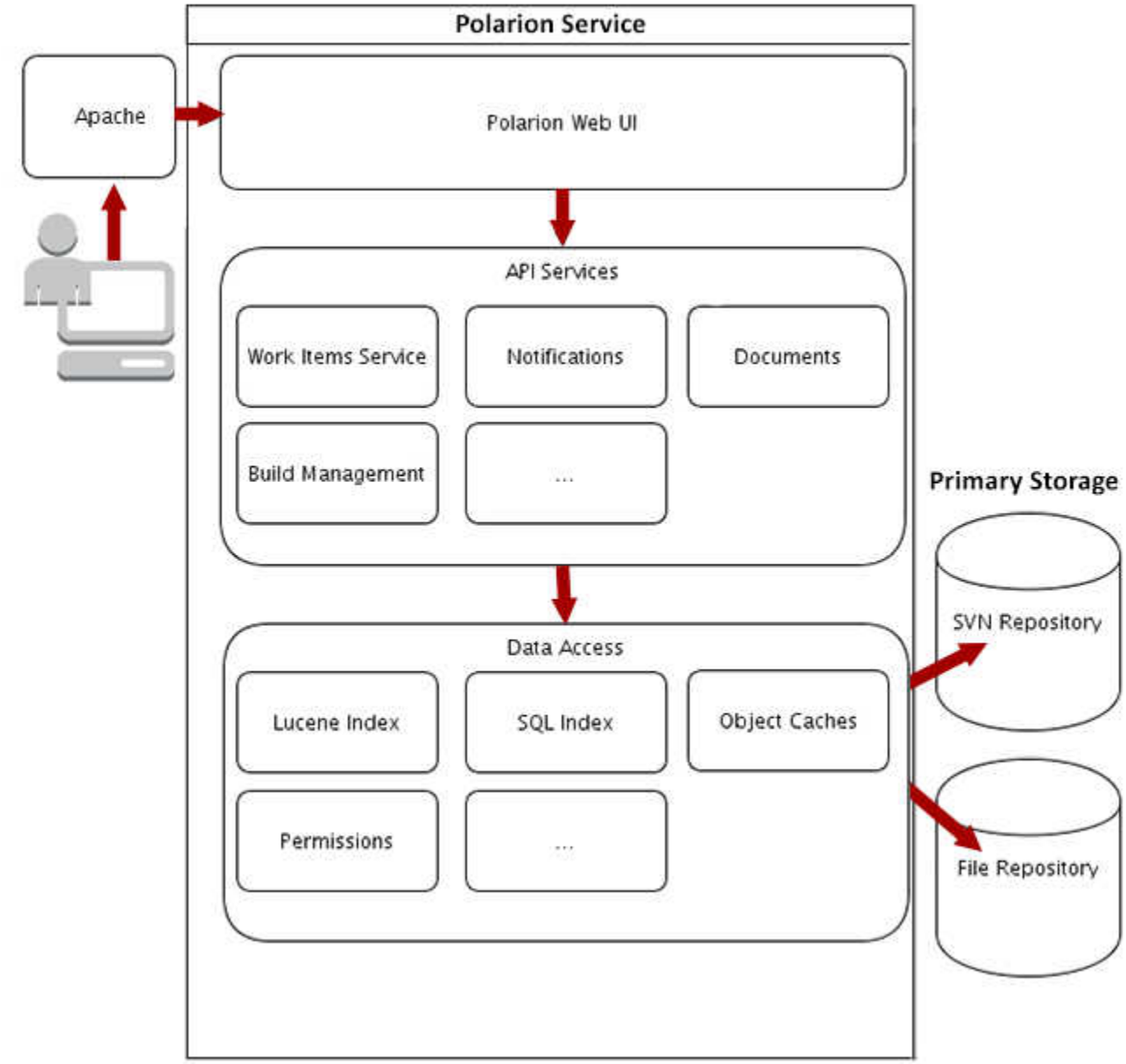
Troubleshooting

- **Linux**: It is recommended to disable SELinux, if it is used.

- **Windows**: Disabling the firewall on enterprise editions of Windows also disables crucial network services.

- After encountering problems with activities, for example, **org.apache.lucene.index.IndexNotFoundException: no segments * file found in MMapDirectory@/opt/polarion/shared/data/workspace/polarion-data/index/Activities**, the index of activities must be manually deleted from the shared folder and a node restarted so that an empty index is created. By default:it is found in the **/opt/polarion/shared/data/workspace/polarion-data/index/Activities** directory.

11. Appendix: Polarion instance architecture



© 2020 Polarion AG