ionir

WHITE PAPER

Containerized Jenkins with ionir

Table of Contents

Jenkins

Jenkins is a powerful open-source Continuous Integration/Continuous Delivery (CI/CD) platform. Running Jenkins in containers dramatically reduces the overhead associated with starting and running instances, making it easy to scale for build, test, and deployment. Using ionir as the data platform for Jenkins on Kubernetes can accelerate the CI/CD pipeline, scale builds and tests across multiple clouds more efficiently, and reduce time to delivery of business applications.

ionir benefits for Jenkins

- **Accelerates application delivery.** ionir enables rapid scaling, optimizes workflows, and reduces errors, speeding the app delivery process
- **Eliminates Multiple Hardware Silos.** ionir shares COTS servers with Jenkins and other Kubernetes apps. In typical configurations, only additional media capacity is required, resulting in significant CAPEX savings over external storage
- **Reduces Administration.** ionir is orchestrated and runs under Kubernetes, and is managed by the same tools that manage applications and Jenkins. This eliminates the need for separate storage and data management, significantly reducing operational complexity and administrative costs

This document describes four use cases associated with Jenkins, and describes the benefits and best practices for using ionir to optimize these use cases.

ionir and Jenkins Use Cases

ionir's storage and data management platform is an ideal solution to deploy containerized Jenkins. ionir simplifies management and ensures data is as agile as applications in the Kubernetes environment. Data of any size can be moved to any location in seconds, and instantly restored (cloned) at 1-second RPOs, significantly accelerating workflows that require data to be preserved and copied.

Running ionir as the data platform for Jenkins under Kubernetes, allows customers to rapidly and horizontally scale their CI/CD pipelines across multiple clusters. ionir provides persistent, resilient, and high-performance storage that reduces run times for builds and tests, in both monolithic and scaled Jenkins environments.

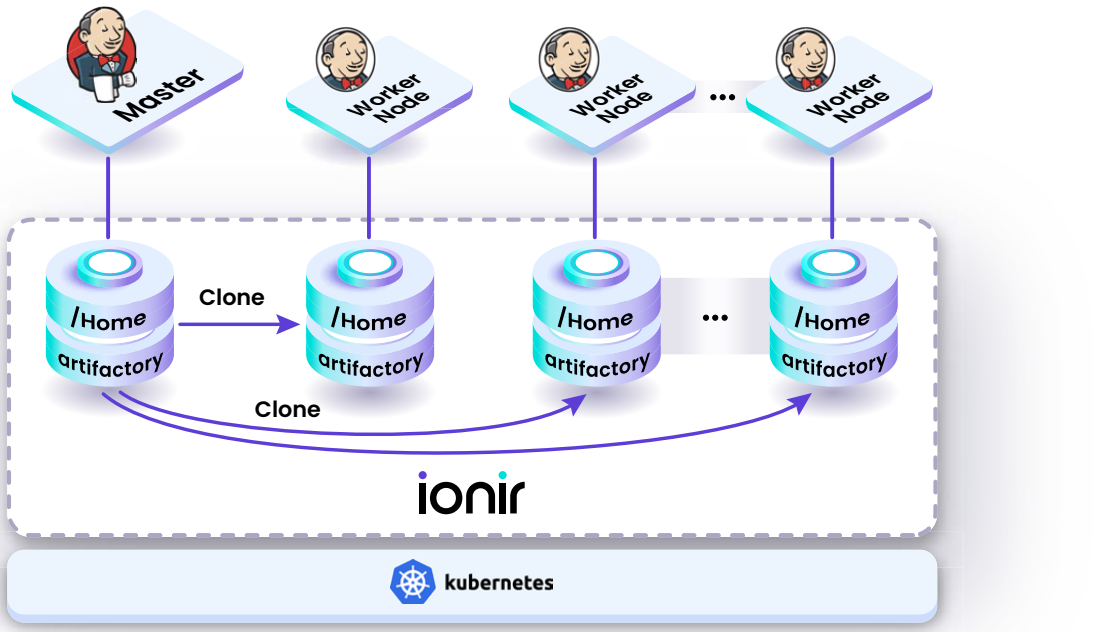
In each of the scenarios covered in this document, the Jenkins declarative pipeline is used to create ionir clones and mount them to Jenkins worker node pods. Each step in a Jenkins pipeline can create a worker pod. The Jenkinsfile of the pipeline holds the configuration required for each step including a pod template of each worker pod. The pod template allows you to set the worker container images and their volumes (Persistent Volume Claim) PVC.

The scenarios that are covered are:

- Horizontal Test Scaling
- Incremental Builds
- Reliable Testing
- Remote Worker Node Scaling

Horizontal Test Scaling

Test scaling expedites time to complete a test run by increasing concurrency, and distributing tests to multiple worker nodes. Rapid scaling requires a simple way to quickly clone the resources (usually the /home directory) and the artifacts and make them available to different worker nodes.



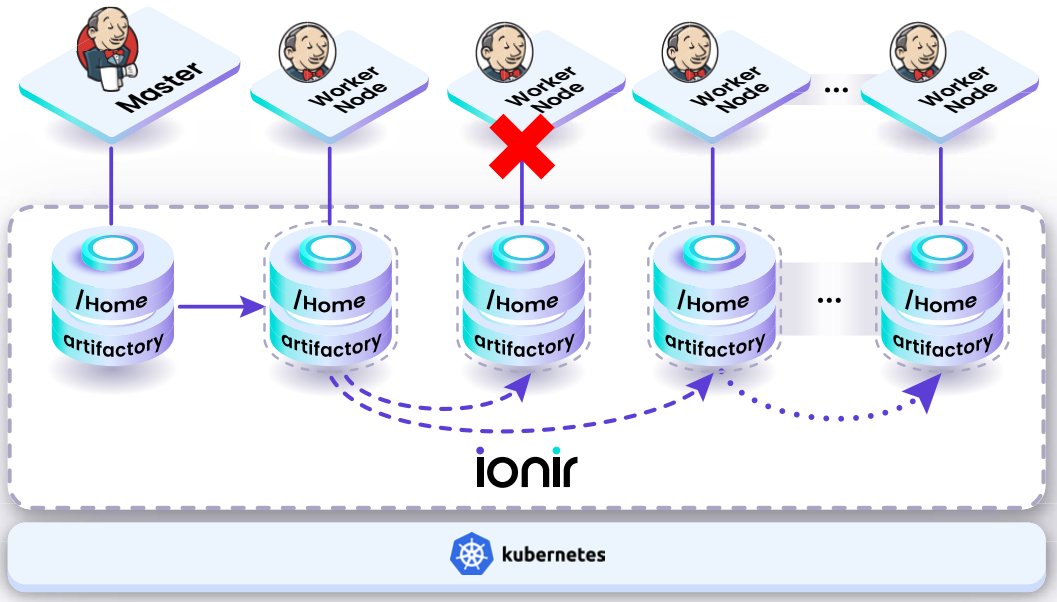
Scaling with ionir is simple. While the worker is created, it can be configured to create an instant clone of the /Home directory and attach it to the worker node.

The following steps must be defined in the Jenkinsfile to create a worker node pod with a clone:

1. Create a Jenkins stage that clones the volume/s of the /home directory and the artifactory (if needed)
2. In the pod template of the next step, use the name of the cloned PVC
3. When the worker node pod starts, it will mount the cloned volume/s
4. At the end of the stage, copy the artifacts that were created if any, back to the Jenkins master and delete the worker node pod and the cloned volumes (PVC)

Incremental Builds

Incremental builds provide the ability to use the outcome of previous builds to test changes quickly. Each build can be the source to a single or multiple build. If a build fails, it should be possible to quickly revert to a previously known good state to avoid the need to run all previous builds.



ionir's ability to instantly create clones simplifies and speeds up Jenkins incremental builds. Each incremental build clones the data and artifacts from the previous build. If any build fails, a simple process recreates the clone from the previous build, fixes the problem, and reruns the build — saving many steps.

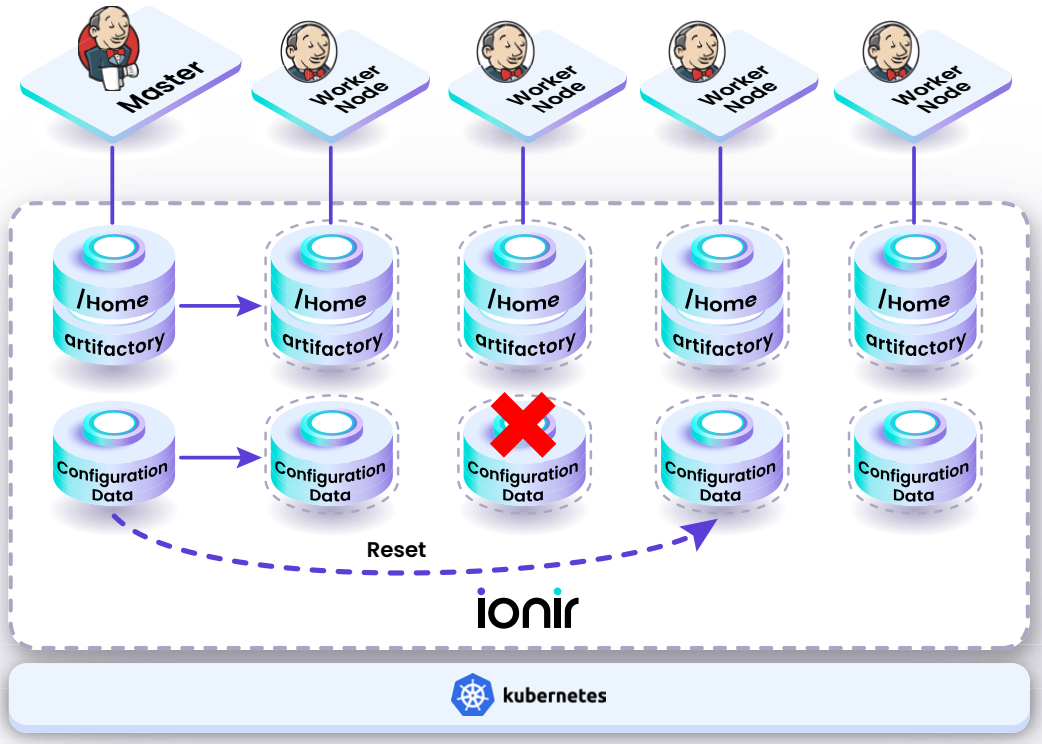
The following steps need to be defined in the Jenkinsfile for incremental builds:

1. Create a Jenkins stage that clones the volume(s) of the /home directory and the artifactory (if needed). In the case of the first build, the clone is done from the master
2. In the pod template, use the name of the cloned PVC(s)
3. When the pod starts, it will mount the cloned volume(s)
4. Did volumes mount successfully?
   - If the step completes successfully: delete the pod, clone the volume/s and move to the next step with the cloned volume
   - If the step fails: Fix the cause of the failure. Re-clone the volume/s from the previous step and rerun the step

At the end of all the incremental builds (steps), copy the artifacts that were created, back to the Jenkins master and delete the worker node pod and the cloned volume(s) (PVC).

Reliable Testing

Many test environments contain configurations and additional data that is used for the various tests. A test can potentially modify this data, and potentially cause subsequent tests to fail. These failures can be difficult to debug thus slowing down the test process.



Using ionir clones to preserve the configuration files and test data provides an easy way to reset the test environment between tests. Doing this provides two benefits:

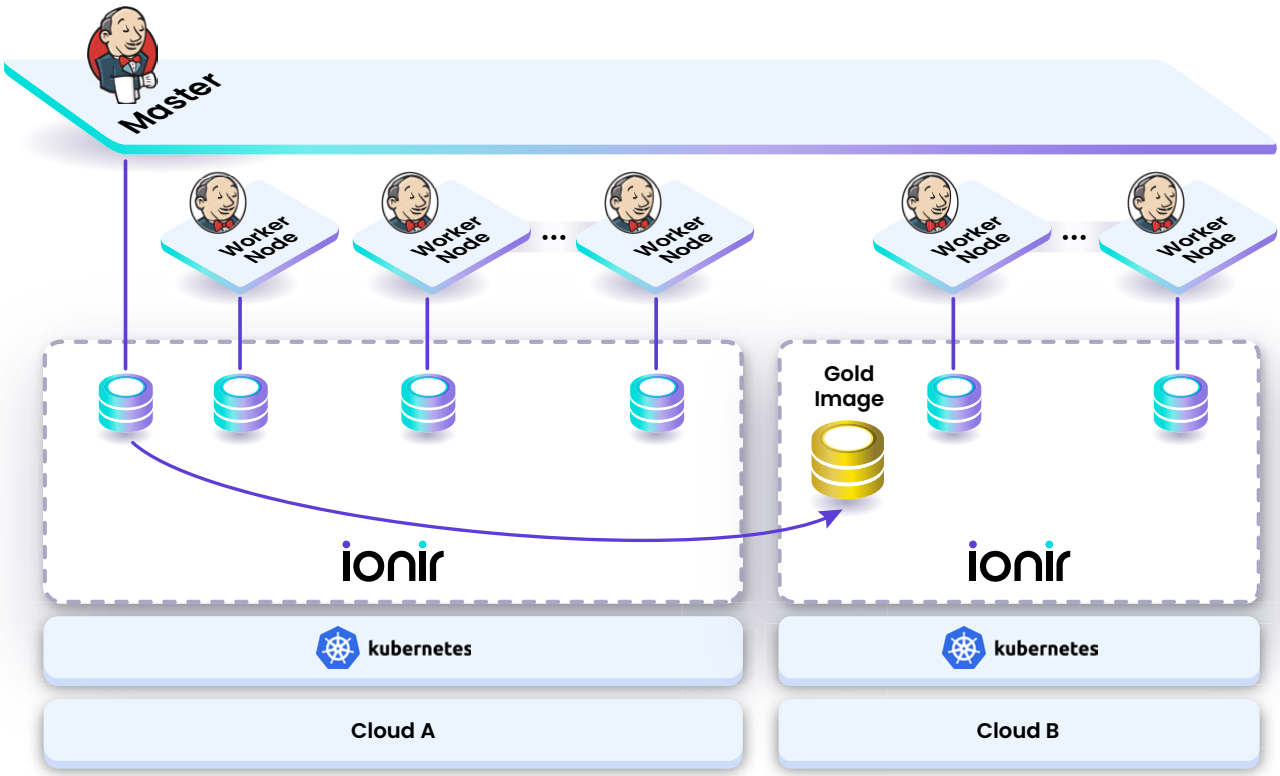
- **Faster test runs:** resetting the data between test runs can reduce the chance of changes to configuration and test data causing test runs to fail. Cloning is instant so there is no delay before running the next test
- **Debug support:** Preserved configuration and data can be used to debug issues that are discovered during the test

Configuration files and any additional data required for the test runs must reside on a ionir volume(s). The following needs to be defined in the Jenkinsfile to support resetting and preserving test configuration and data using ionir clones:

1. Create a Jenkins stage that clones the volume(s) of the configuration files and data
2. In the pod template for worker node, mount these volume/s using the name of the cloned PVC
3. Run the test and once the test completes, the environment can be reset by deleting the clones and creating new clones for the next test
4. In case of test failures, the original clone(s) can be preserved for debug

Remote Test Scaling

Remote scaling keeps a single Jenkins environment (master) that spans multiple clusters / clouds and can be valuable in hybrid and multi-cloud environments.



Remote scaling benefits from ionir's ability to instantly copy data of any size, anywhere. This powerful feature means that copied data is accessible at the new location in seconds, and can be used immediately for tests and builds. Additionally, clones can be created at the remote location to further support the three use-cases discussed above.

Conclusion

ionir's storage data management platform is an ideal solution to deploy containerized Jenkins. Data agility, including the ability to move any size volume to any location in seconds, and the ability to instantly restore 1-second RPOs significantly accelerates workflows that require data to be preserved and copied.

ionir

info@ionir.com | ionir.com