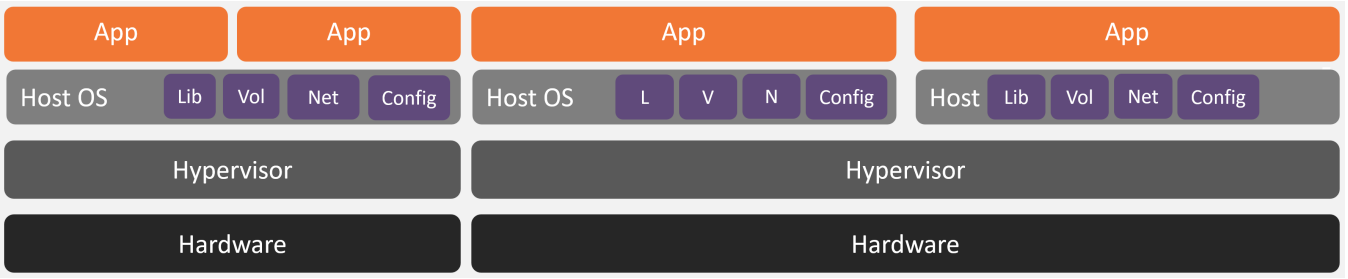
jtel Container Stack - Introduction

Containers are a way to package an application together with everything it needs to run (runtime, libraries, system tools, config), and execute it in an isolated environment on a shared operating system kernel.

Unlike a virtual machine, a container **does not include a full operating system**. It runs as a normal process on the host, isolated from the host and the other containers, sharing the single OS kernel from the host.

Old School Approach

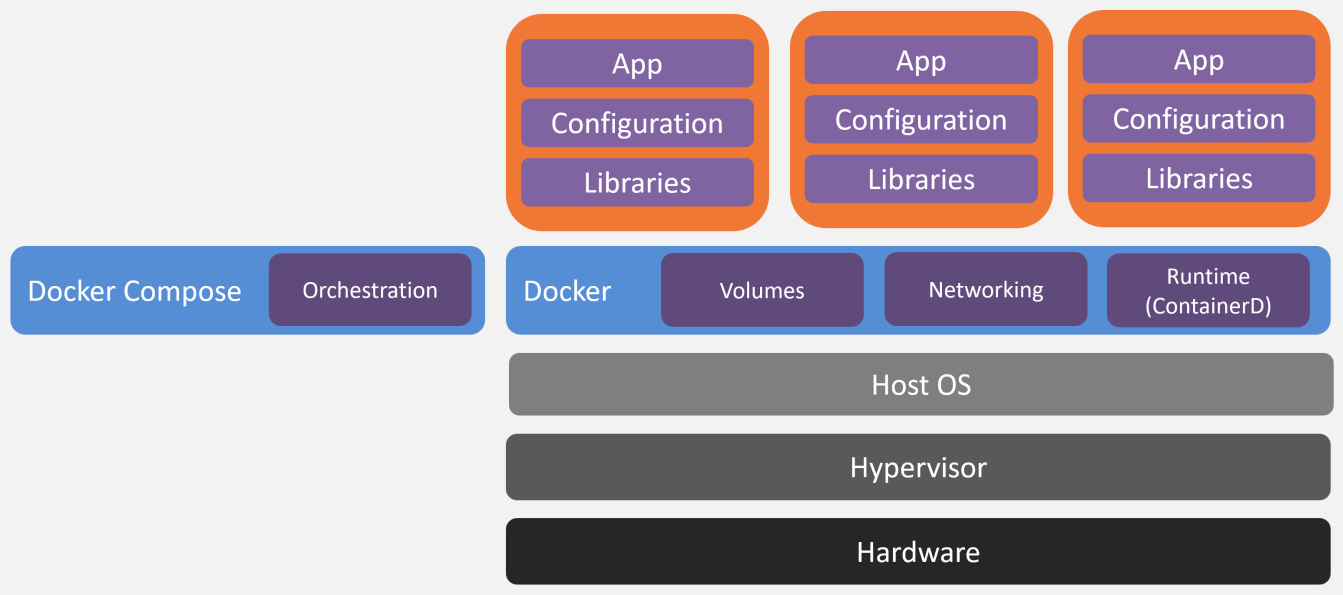


Disadvantages:

- High resource overhead
- Slow startup
- High maintenance cost (DevOps operation costs like installing / patching / maintaining many OS and applications)
- Murky Development and DevOps boundaries - "Works on my machine" problems

Container Approach

The main difference between container deployments seen in the industry is the orchestration (i.e. what is responsible for starting and stopping the containers amongst other tasks). The jtel container stack uses **docker compose**.



Advantages:

- Lightweight - low resource overhead
- Much higher density per host
- Fast startup in seconds or less
- Maintenance like Updates, smaller patches or migrations streamlined and efficient
- Fast Desaster recovery (The complete stack can be built from the ground up in minutes, depending mostly on the database size for speed of deployment)
- Clean Devs / DevOps Boundary

One Process for all Scenarios

Our container stack uses the same process for all possible scenarios:

- A new vanilla (empty) installation
- Migrating from an existing old school installation to a container installation
- Migrating from one container stack to another stack hosted somewhere else
    - Change IaaS providers
    - Out of place updates
- Disaster recovery from the daily backup

This means, that we effectively test disaster recovery every time a stack is deployed or updated.

Involved Repositories and Storage

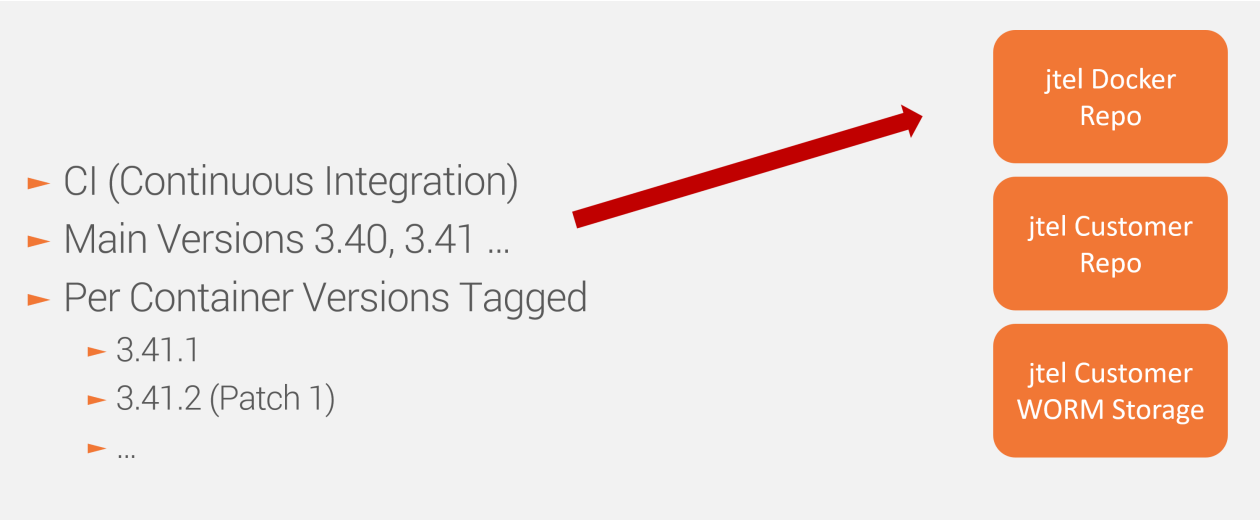
There are four main repositories and storages involved in the jtel Stack Container deployment process:

- jtel Docker Repo
- jtel Customer Repo
- jtel Customer WORM Storage
- jtel WORM Storage

jtel Docker Repo

This is the jtel repository in which all container images are stored. Container images are tagged every time a version is produced, for example:

- 3.43.1 - main version, first release
- 3.43.2 - main version, first patch release

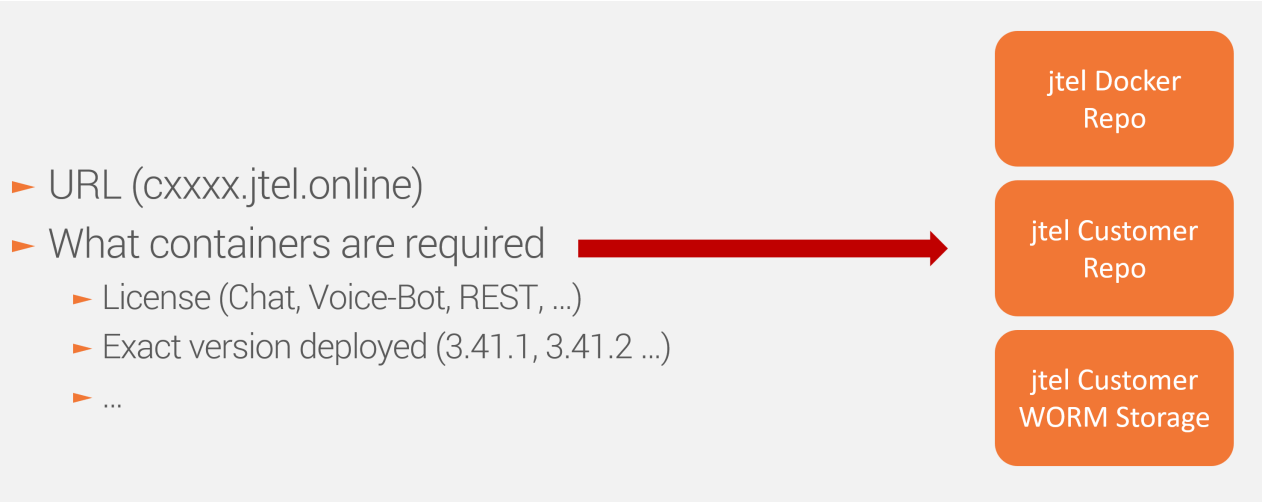


The required images are built automatically built and tagged by the jtel CI Pipeline every time a developer merges a feature to a particular release branch.

jtel Customer Repo

This is a customer (jtel container stack instance) specific repository which contains:

- The exact configuration for the customer
  - What exact version is used (main release and patch tag)
  - Which containers are activated
  - What is the sizing of the stack (Small / Medium / Large)
  - DNS names
  - Container host specific configuration
- Whenever a change is made for the customer, it is documented in this repo for future reference



Modifications to the jtel Customer Repo are logged, so changes can be tracked easily.

It is possible to go back to a previous incarnation of the stack / installation as needed.

All Backup Scenarios

None of the backup scenarios described provide:

- Backup of call recording audio files and voice recordings
- Backup of detailled call and process logs

If you require retention of call recording audio files or voice recordings, then you must provide an FTP storage to which call recordings and voice recordings will be uploaded.

Detailled call and process logs are not be backed up in any scenario.

jtel Cloud SaaS Customer WORM Storage backup

This applies only for jtel Cloud SaaS systems. For On Premise installations, the backups are stored and maintained by the customer. See below.

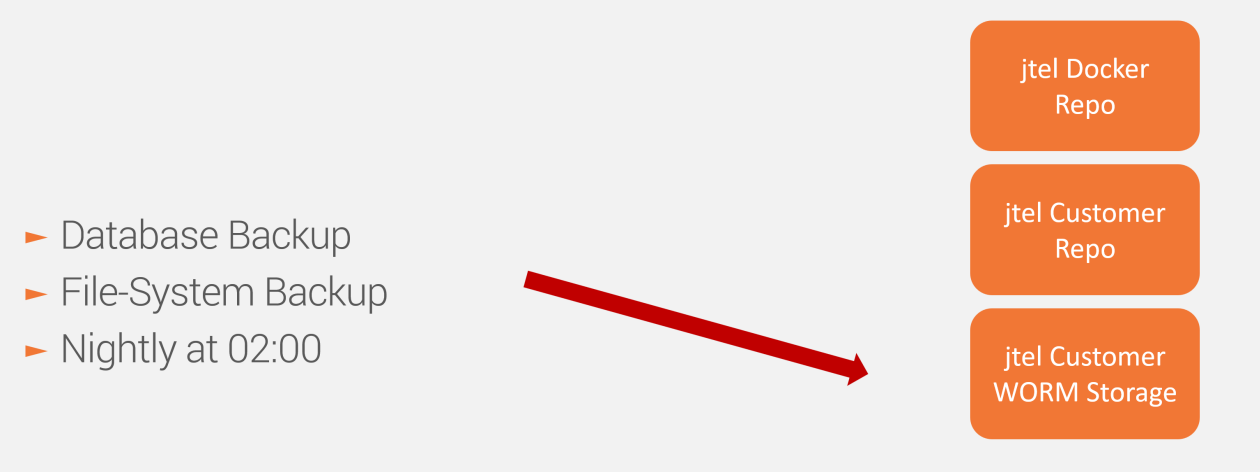
The system makes a backup to WORM storage (Write Once Read Many times) which is hosted by jtel in Azure (Geo Location: Germany West). The backup is created once a day at 02:00.

The backup is a content based backup which allows for re-creation of the same container stack on any suitable host.

The retention policies ensure that:

- The storage account cannot be deleted, even by global jtel admins for 180 days
- The backups are immutable and therefore cannot be encrypted

For initial stack setup, a vanilla backup provided from the jtel WORM Storage is used.



The backups from a running system are made nightly, automatically, by the acd-task-runner container.

Each WORM storage is a specific storage instance housed in a customer specific storage account and is used for one customer only. It has it's own unique access credentials and endpoint providing a very high level of security. The upload is done via rclone command Networking related information here

Cloud

jtel Container Stack - Networking Cloud

OnPrem

jtel Container Stack - Networking OnPrem

jtel WORM Storage

This immutable storage contains version specific files and data stored for specific releases. These files are required and copied to the customer specific storage when an initial (empty) installation is made or when an update to a new release is performed.

The jtel CI Process uploads new files and data here automatically from the build process.

This applies only for jtel Cloud SaaS systems. For On Premise installations, the backups are stored and maintained by the customer. See below.

jtel Container On Premise Backup

For On Premise installations, the customer provides jtel with a fileshare accessible to the container system on the network (SMB or NFS). The jtel Container System mounts this fileshare in the Container stack.

Automatic backups are created once a day at 02:00.

The backup is a content based backup which allows for re-creation of the container stack on a suitable host.

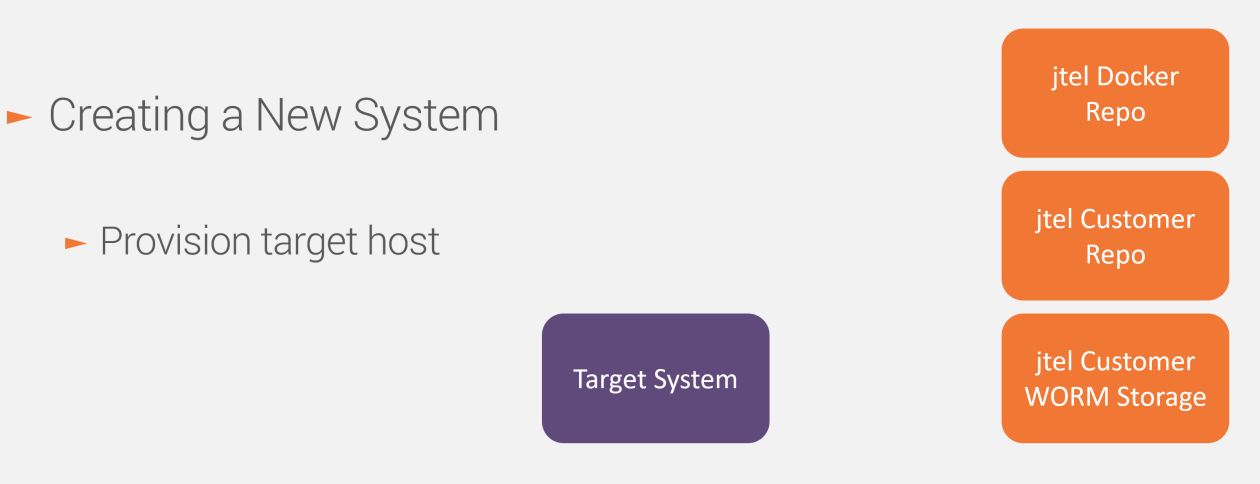
For initial stack setup, a vanilla backup provided from jtel is used.

The customer is responsible for implementing suitable backup retention periods.

Creating a new Installation

Manual: Target Host Provisioning

The target host is created. This will usually be a operation which a DevOps person will trigger manually.



Manual: Customer Repo Configuration

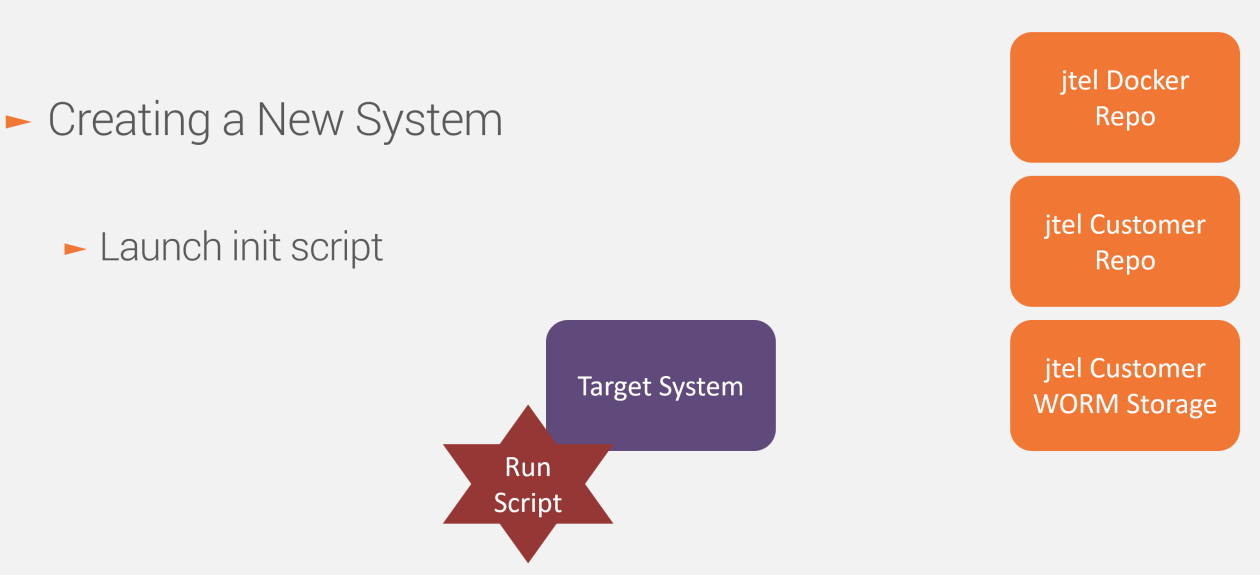
The system configuration is entered to the customer repo according to what is ordered. This step will be manually completed.

This defines:

- what containers
- what sizing
- DNS names
- the target host machine environment

Manual: Start Script on Target Host

This is a one-liner which can be run from the DevOps machine remotely via SSH (assuming SSH access is available) or directly on the container host itself.

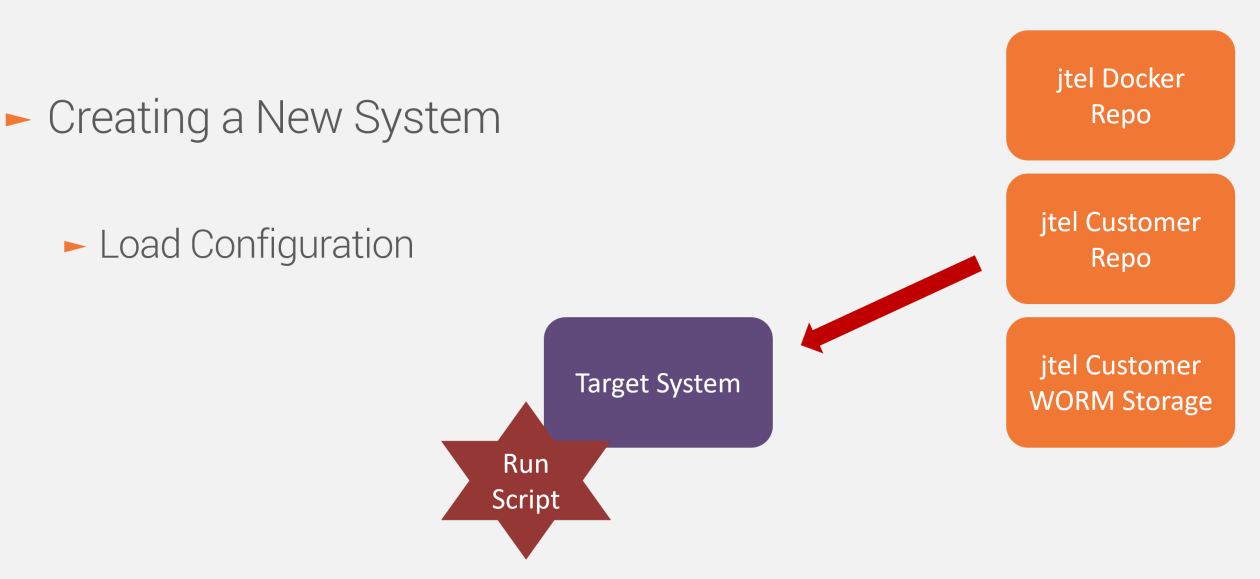


Automatic: Automatically Copy Vanilla Backup

If a backup is not already provided (which is the case for new systems), then the init script copies retrieves the vanilla backup to the customer WORM storage.

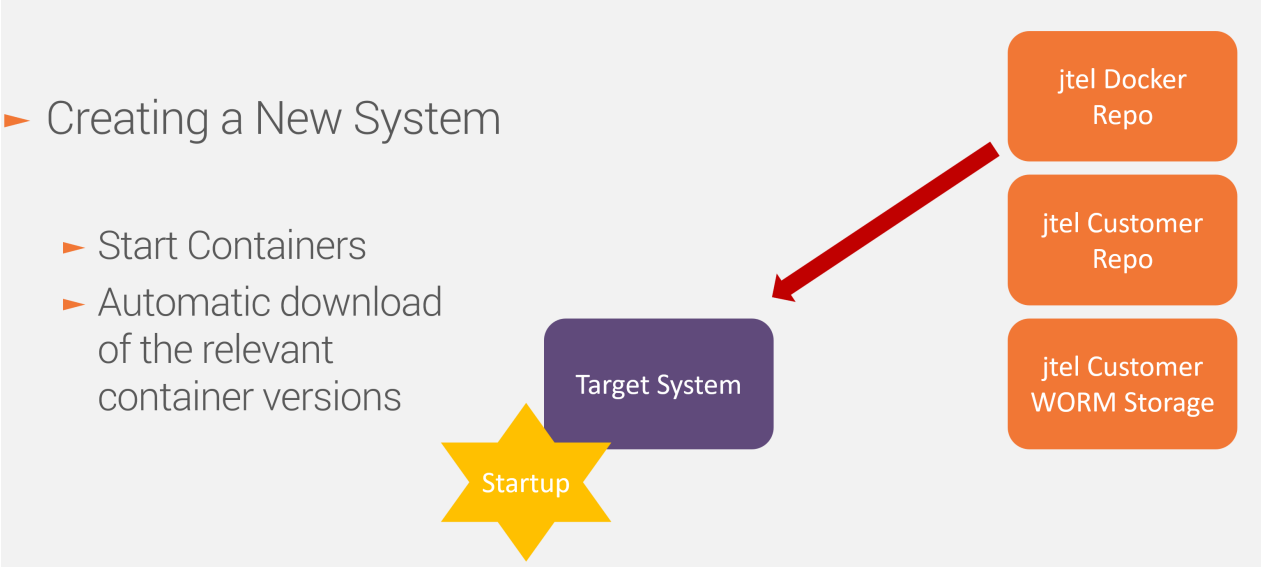
Automatic: Load Configuration

The system automatically loads the configuration from the customer repo. The host will be configured at this stage to match certain settins, such as DNS.



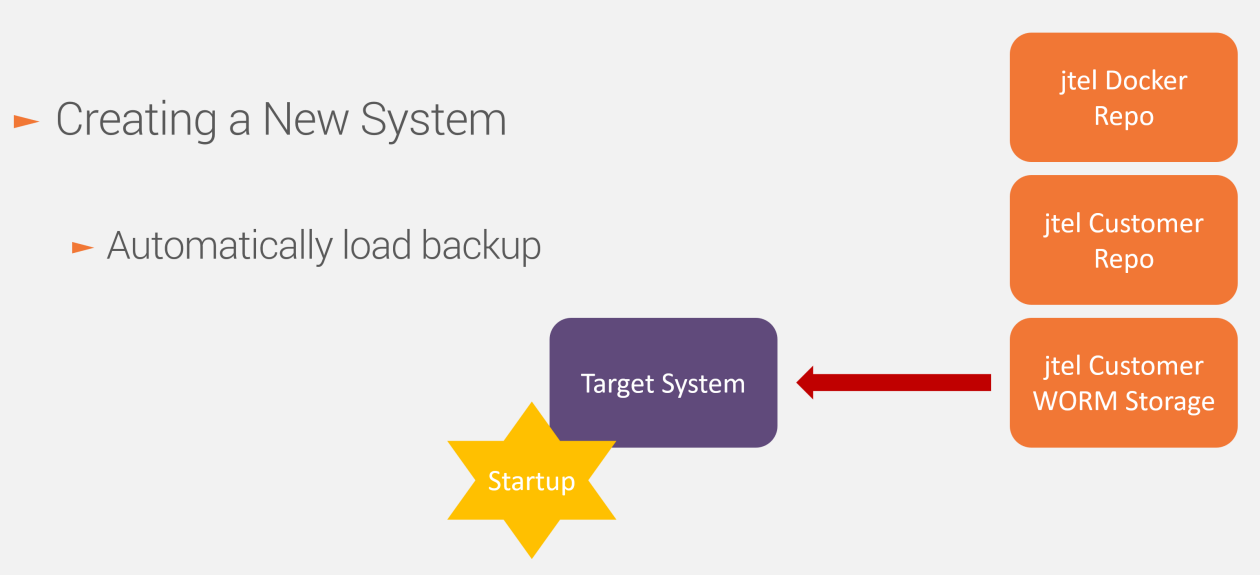
Manual: Start Containers

This is a one liner. The containers will automatically be downloaded in the correct version.



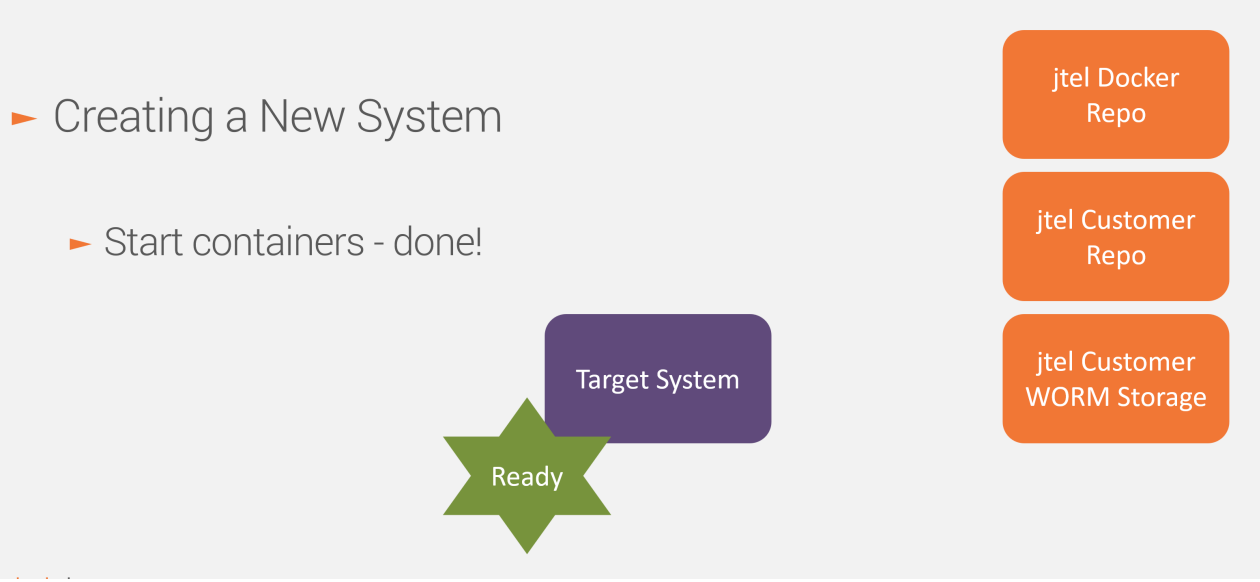
Automatic: Load Backup

The system loads the backup into the database and file system automatically if it is empty.



Automatic: Perform Updates if Needed

The containers are started up and these automatically perform any updates necessary (for example to the database) required by the selected version and patch release.



Updating a System - In Place

Updating a system in place requires the following steps:

- The version to update to is configured in the customer repository
- The customer repository is updated on the target host (one liner)
- The following steps are the same as above from "Start Containers"
- When the containers are started, the required versions will automatically be pulled and any updates necessary will be applied to the database and file system.

Updating a System / Disaster Recovery / Moving a System - Out of Place (Migrations)

The same procedure is used for all of the following use cases:

- Disaster Recovery
- Migration / Moving a System
- Out of Place Updates

This requires the following steps:

- The version to update to is configured in the customer repository if an update is to be performed
- The new host details are entered to the customer repository
- The following steps are the same as above from "Start Script on Target Host"