

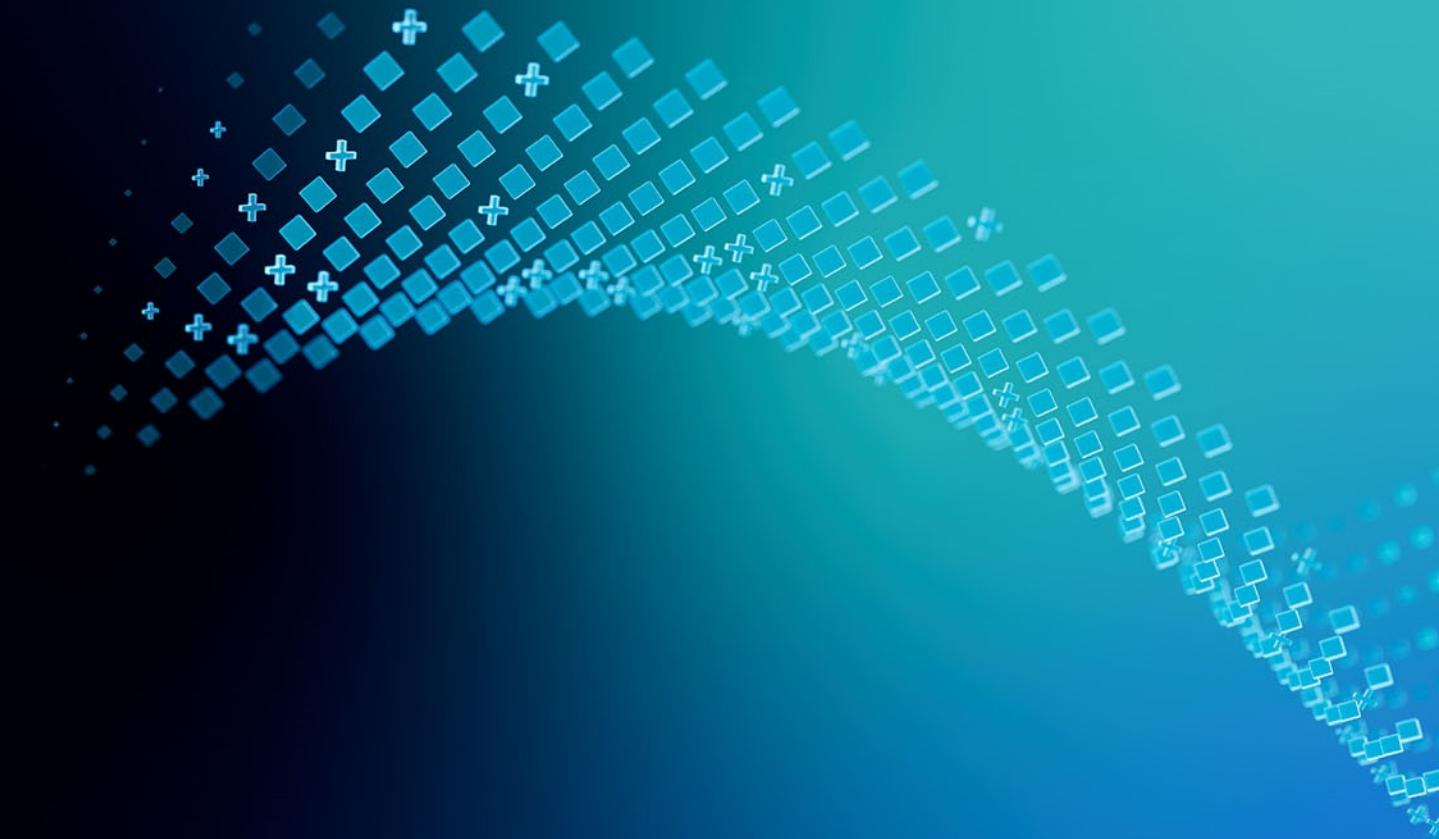


# Domino on Kubernetes

DACHNUG, June 2023



Daniel Nashed -- <https://blog.nashcom.de>

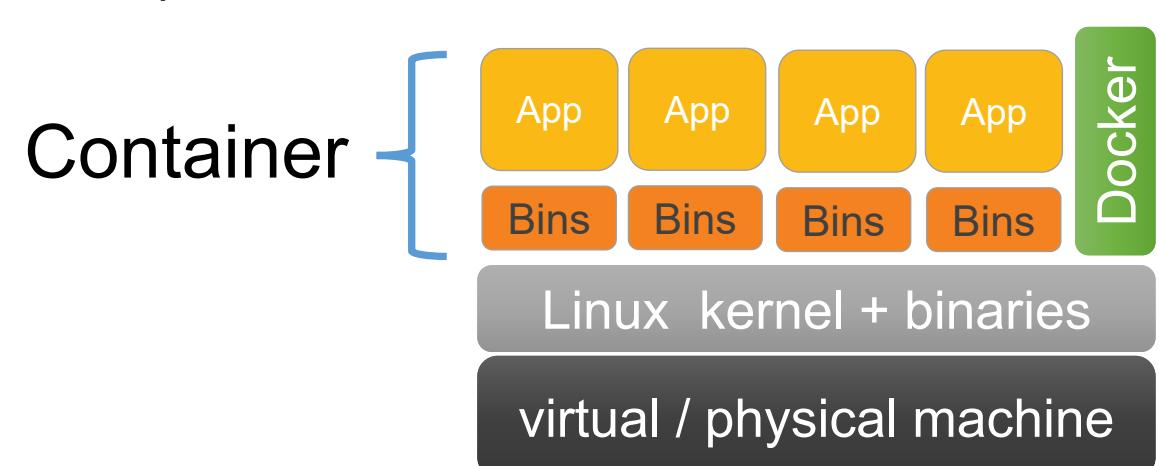


## Running Domino in Containers on Docker & Kubernetes



# Reasons to use Container Technology

- **Automation testing**
- **Continuous development & deployment**
  - e.g **consistent deployment** of applications in **test**, **QE** and **production** environments
- Cost reduction by **standardization & automation**
- Cost reduction by **lowering the overhead** to deploy and run a Container vs. VM
  - Application runs an easy to deploy **container** instead of a separate **virtual machine**
- **Automated updates** with less down time  
(for individual servers)
- **Containers can improve security**
  - If built and deployed in the right way!



# Containers in Automation Testing



- Continuous development requires continuous automation testing
- HCL Domino development team uses automation testing for every git submit
  - All automation tests have to pass as one requirement before code can be committed!
- More complex integration tests are running every daily build
- New technology requires test suites integrating with other systems (OIDC, Let's Encrypt, ..)
  - Leveraging containers new automation tests
  - New HCL Domino 12.0.2 container image is used for all container based automation testing



# Domino Container Image Test Automation

- The Domino container image project provides an automation testing suite
- Part of the **Open Source GitHub project**
  - Extendable by customers and partners
  - The same test you find in GitHub repository is running in a **Jenkins pipeline** every night on the daily build
  - Leverages **OneTouch setup**, results are written in a standardized **JSON** format

```
{  
  "testResults": {  
    "harness": "DominoCommunityImage",  
    "suite": "Regression",  
    "testClient": "testing.notes.lab",  
    "testServer": "testing.notes.lab",  
    "platform": "CentOS Stream 8",  
    "testBuild": "12.0.1FP1",  
    "containerPlatform": "Docker",  
    "containerPlatformVersion": "20.10.21",  
    "kernelVersion": "4.18.0-408.el8.x86_64",  
    "kernelBuildTime": "#1 SMP Mon Jul 18 17:42:52 UTC 2022",  
    "glibcVersion": "glibc-2.28-214.el8.x86_64",  
    "libstdcVersion": "libstdc++-8.5.0-15.el8.x86_64",  
    "testcase": [  
      {  
        "name": "domino.jvm.available",  
        "description": "Domino JVM available",  
        "executionResult": "SUCCESS",  
        "errorText": ""  
      },
```

```
[ SUCCESS ] domino.jvm.available  
[ SUCCESS ] domino.server.running  
[ SUCCESS ] domino.http.running  
[ SUCCESS ] domino.certificate.available  
[ SUCCESS ] domino.server.onetouch.microca-cert  
[ SUCCESS ] domino.server.onetouch.createdb  
[ SUCCESS ] domino.idvault.create  
[ SUCCESS ] domino.backup.create  
[ SUCCESS ] startscript.archivelog  
[ SUCCESS ] container.health  
[ SUCCESS ] startscript.server.restart  
[ SUCCESS ] domino.translog.create  
[ SUCCESS ] domino.smtp_pop3.mail  
  
-----  
Success : 13  
Error   : 0  
Total   : 13
```

# Domino Container Images



- **HCL Image**
  - Download HCL Domino 12.0.2 Container from **Flexnet** or **pull from HCL registry**
  - Documentation is updated, including examples
    - [https://help.hcltechsw.com/domino/12.0.2/admin/inst\\_dock\\_deploying\\_new.html](https://help.hcltechsw.com/domino/12.0.2/admin/inst_dock_deploying_new.html)
- **HCL Community image**
  - Built by customers & partners
    - Download **Linux WebKit installer** and just run “**./build.sh domino**”
    - Support for the build process and the additional image functionality is **GitHub community based**
- Support for **standard Domino & start script functionality** in both containers by HCL
  - Additional functionality is supported by the community project via GitHub
  - **Kernel of host + glibc & libstdc++** version of the container must match system requirements!



# Domino Container Community Image

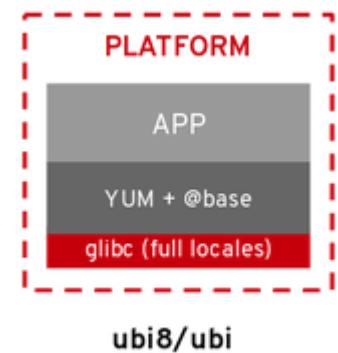
- Domino Docker “image” available since Domino 9.0.1
  - Originally initiated at IBM as a GitHub Open Source project by Thomas Hampel
  - Moved to HCL <https://github.com/HCL-TECH-SOFTWARE/domino-container>
- Not a ready to use image to download
  - But you get a nice box of “**LEGO® bricks**” with a lot of flexibility
  - **Easy to use build script** leveraging HCL Domino Linux server web-kits
  - Provides download links for HCL **web-kits** from FlexNet and download hash checking
- Additional functionality & flexibility
  - Add-on installers for **Nomad Server**, **Verse**, **C-API** build environment, **Borg Backup**, ...
- Supported by community project on **GitHub**



# New Domino 12.0.2 Container Image



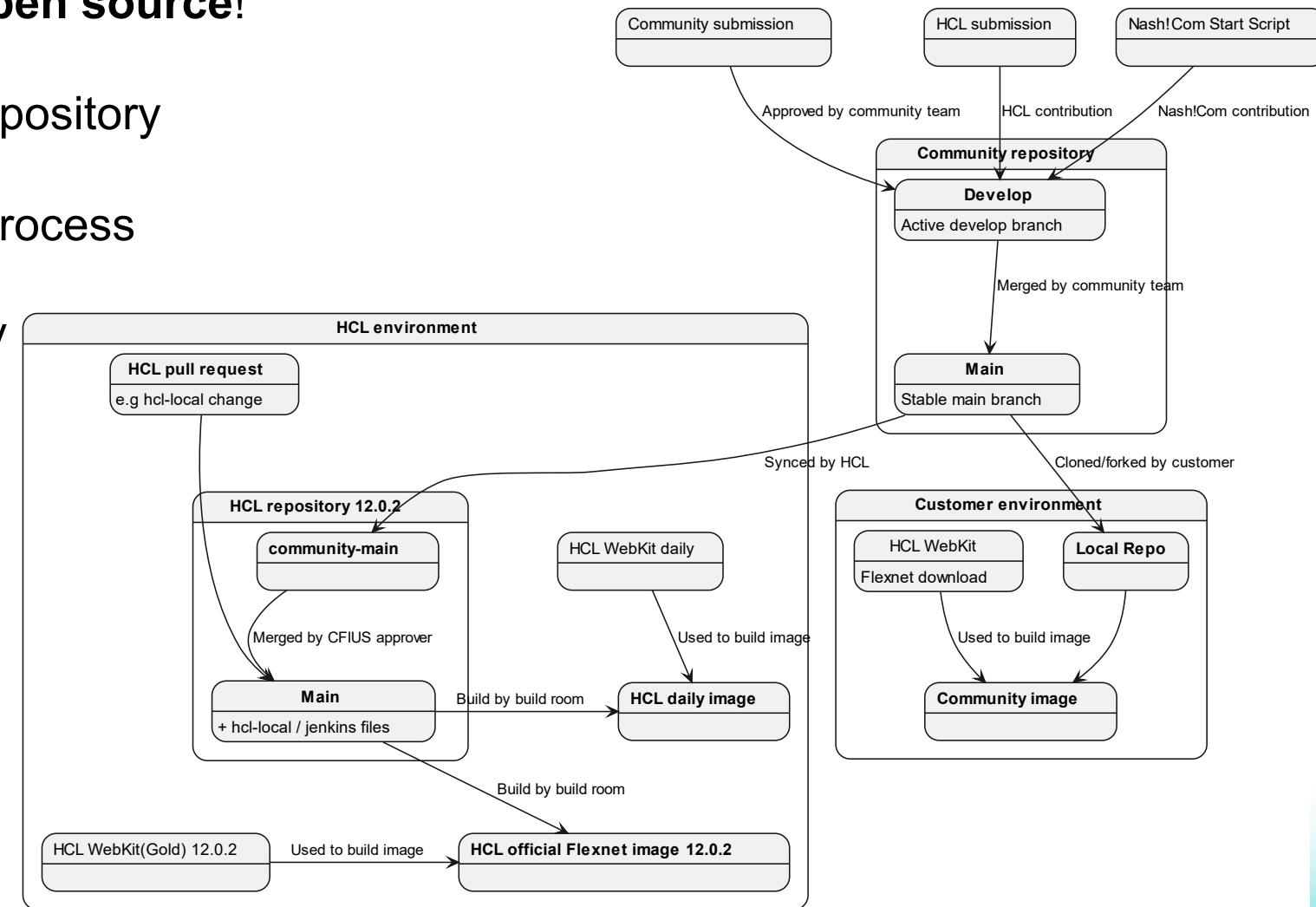
- Based on Community image with all it's included image functionality
- **Stable snapshot of the community project**
  - Updates are planned for each release / fixpack
- Supports the functionality of the community image without the additional components
- Special build of the community image
  - **RedHat Universal Base Image (UBI) 8.6**
  - Only required Linux packages are installed (e.g. no OpenSSL and other helper tools)
- New **HCL Traveler 12.0.2 & Domino Leap 1.1** image are based on new image



# HCL GitHub Project Setup



- Project continues to be developed **open source!**
- HCL has an own internal “synced” repository
- Internal development and approval process
- Build tools in `/hcl-local` directory
  - Jenkins scripts for build room
  - Security scan configuration
  - ...



# Changes to previous Image



- Most of the basic functionality stays the same!
- No more temporary containers for **setup, maintenance and update**
- Admins just initiate a new container to setup or update with a new image
- **Migration from the previous image to the new image is just running a new container!**
- Domino Server process output is written to `/local/notesdata/notes.log`
  - The `notes.log` will be compressed automatically on restart
  - The previous container image wrote all the output to `stdout`

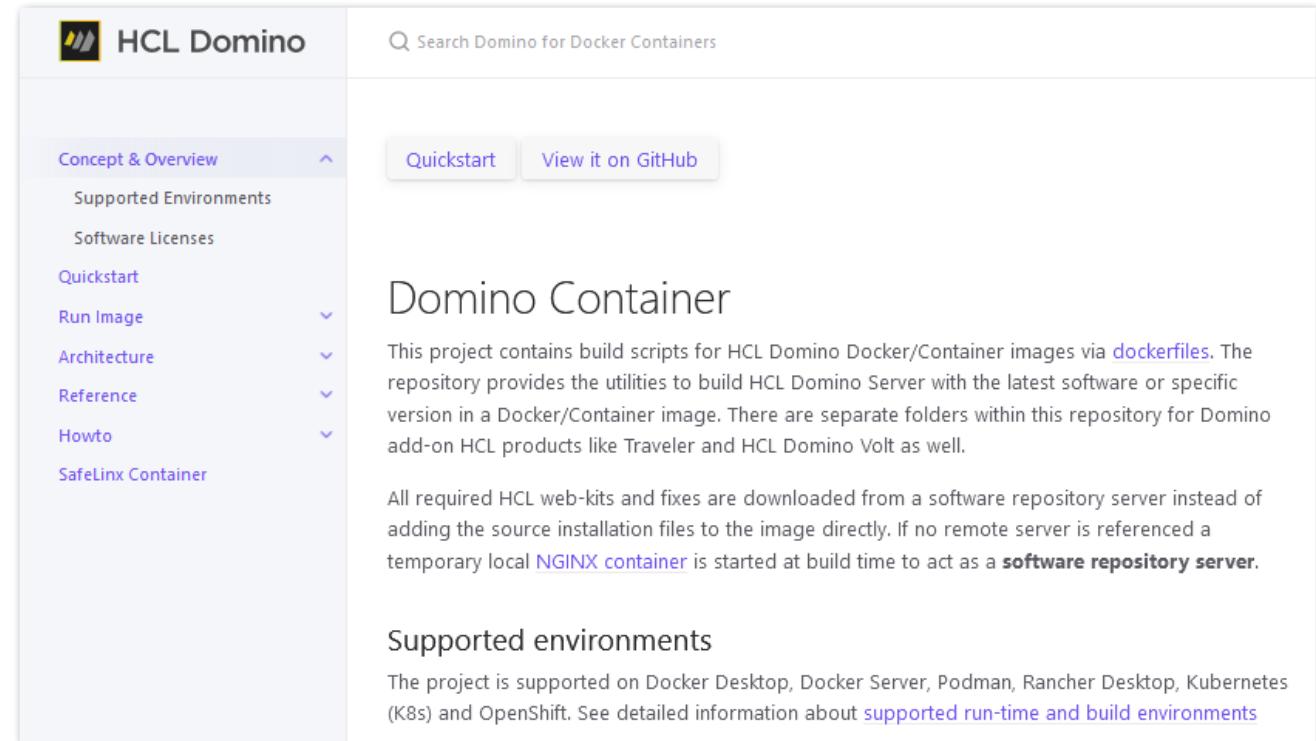
# Domino Community Project & Documentation

- Project & Documentation
- Useful additional documentation
  - Architecture
  - How-To information
  - References
  - Examples



<https://github.com/HCL-TECH-SOFTWARE/domino-container>

<https://opensource.hcltechsw.com/domino-container/>



The screenshot shows a documentation page for the HCL Domino Docker Container. At the top left is the HCL Domino logo. To its right is a search bar with the placeholder "Search Domino for Docker Containers". Below the search bar are two buttons: "Quickstart" and "View it on GitHub". On the left side, there is a sidebar with a tree-like navigation menu under the heading "Concept & Overview". The menu items include "Supported Environments", "Software Licenses", "Quickstart" (which is expanded), "Run Image", "Architecture", "Reference", "Howto", and "SafeLinx Container". To the right of the sidebar, the main content area is titled "Domino Container". It contains text explaining that the project uses build scripts via [dockerfiles](#) and provides utilities to build HCL Domino Server in a Docker/Container image. It also mentions separate folders for add-on HCL products like Traveler and HCL Domino Volt. Below this, another section titled "Supported environments" discusses the supported run-time and build environments, mentioning Docker Desktop, Docker Server, Podman, Rancher Desktop, Kubernetes (K8s), and OpenShift.

# Domino Start Script Project & Documentation

- Project & Documentation
- Full start script documentation
  - Including all commands
- OneTouch Setup information
- Domino Container Control (**dominoctl**)



<https://github.com/nashcom/domino-startscript>

<https://nashcom.github.io/domino-startscript/>

The screenshot shows a website for the Domino Start Script. At the top left is the Nash!Com logo. A search bar at the top right contains the placeholder "Search Domino Start Script". Below the header is a navigation menu with a dropdown for "Domino Start Script" containing links like "Quickstart", "Commands", "Configuration Parameters", etc. To the right of the menu are two buttons: "Quickstart" and "View it on GitHub". The main content area has a section titled "Introduction" which includes a detailed description of the script's purpose and features. Below the introduction is another block of text providing specific setup details.

**Domino Start Script**

- Quickstart
- Commands
- Configuration Parameters
- Components of the Script
- Systemd configuration
- Known Issues
- Domino Container Support
- Domino One-Touch Installer
- One-Touch Domino Setup
- Domino container control
- Fail2Ban for Domino
- Changes

**Introduction**

The Domino cross platform start/stop and diagnostic script has been designed to unify and simplify running Domino on Linux and UNIX. The start script is designed to be "one-stop shopping" for all kind of operations performed on the Linux/UNIX prompt. The script can start and stop the server, provides an interactive console and run NSD in different flavors. It ensures that the environment is always setup correct and supports multiple partitions.

This script is designed to run with a dedicated user for each partition. It is already configured for the default user "notes" user and group (notes:notes). Binary are assumed to be in /opt/hcl/domino, the data directory is assumed in /local/notesdata. Settings are configured in /etc/sysconfig/rc\_domino\_config.

## Docker vs Kubernetes



# Docker vs. Kubernetes



Easier local storage

Easier networking

Lower security requirements

Easy operations

*“Developer friendly”*



# kubernetes

Storage classes, PVCs, ...

Ingress, NodePorts, LoadBalancers

ServiceAccounts, RBAC, ...

Helm, upgrades, monitoring, ...

*“Machine friendly”*

# The basic flow (for both Docker and Kubernetes)



You start with an image. It is like a *template*.

From the image you create a container. It is like an *instance*.

You use parameters to create multiple containers from one image.

A K8s pod usually has one of more containers. Pods are a wrapper around containers to manage them

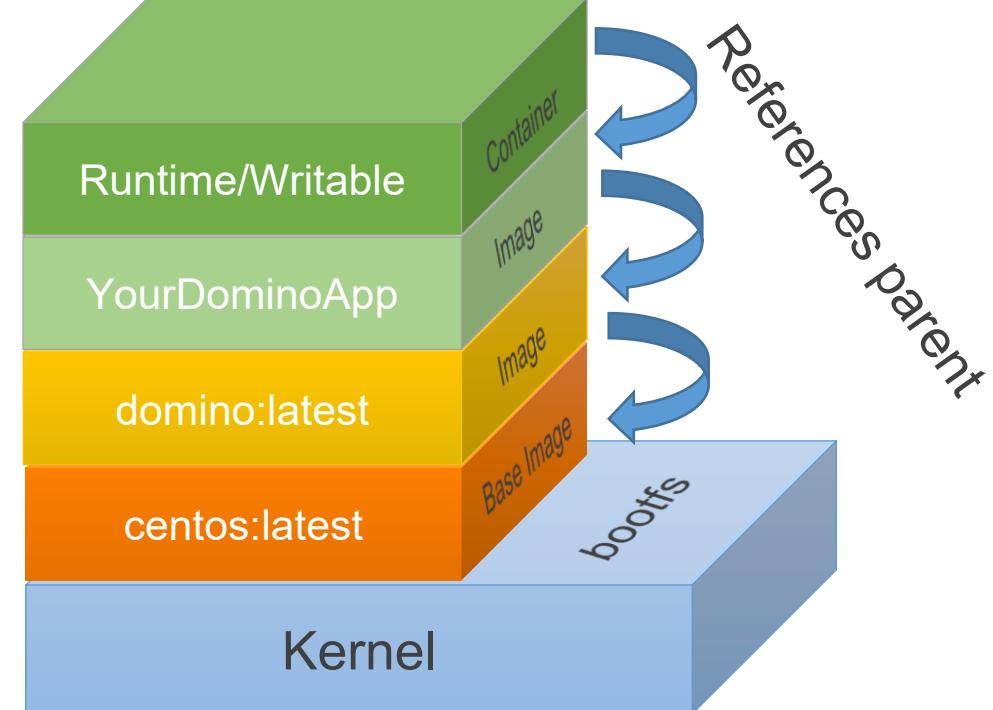
# Container Concept – Example : Domino

## Container is

- a layered file system where each layer references the layer below
- a runtime instance of an image.
- not containing your persistent (Domino) data

## Images

- Are used to create containers
- Layers build on top of each other  
only the differences are stored in each layer.





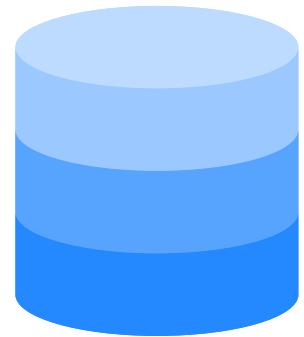
# Binaries and data split

Necessary for running inside a container.



[`/opt/hcl/domino`](#)

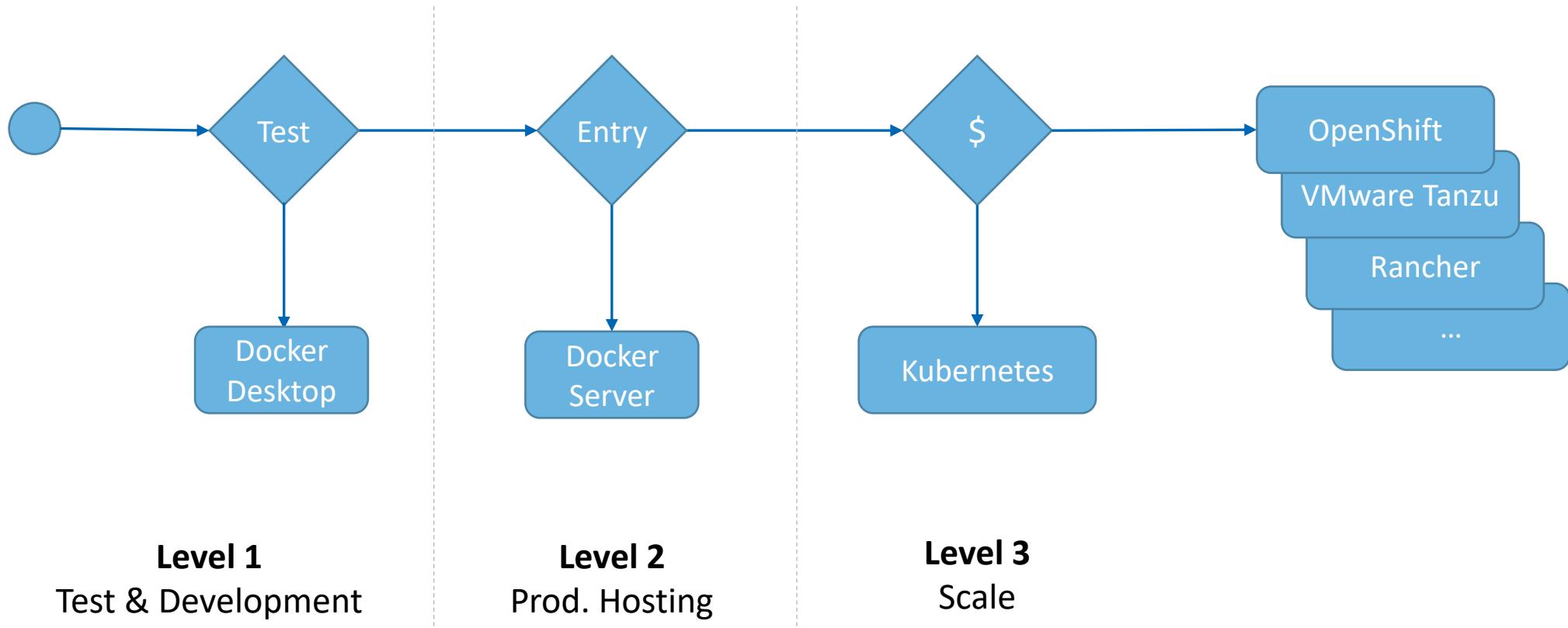
Domino binaries  
Traveler, Verse,  
Leap, Nomad, ...



[`/local/notesdata`](#)

`*.nsf, *.ntf, *.id,`  
`notes.ini,`  
`translog, daos, ...`

# Where to start - Decision Flow



# Container Terminology for Domino Admins

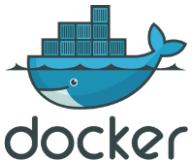


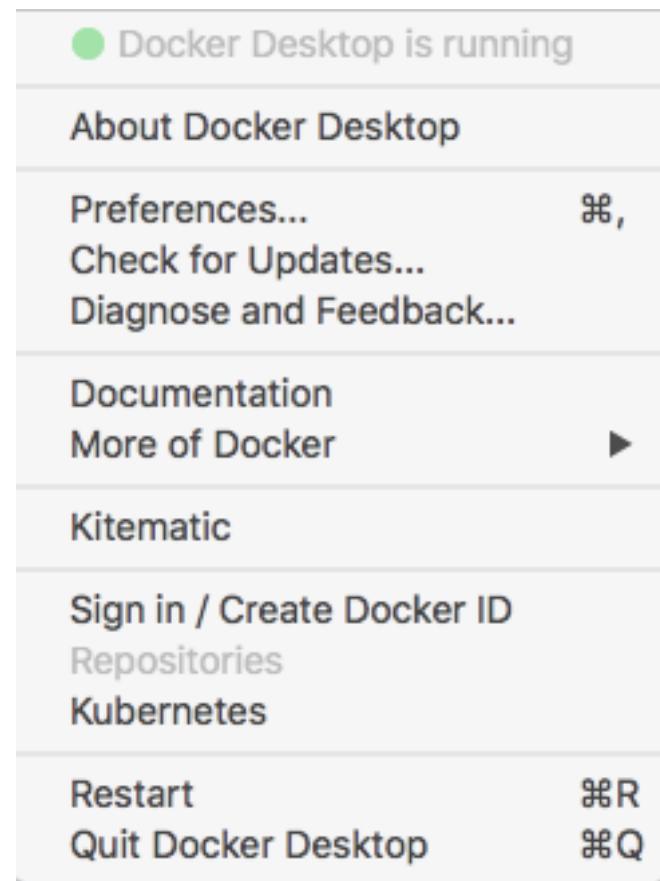
Image	=	Template
Container	=	Application
Volume	=	Domino Data Directory
CLI	=	Domino console
Registry	=	OpenNTF.org

# Level 1 - Docker in MacOSX

- Uses Mac integrated virtualization
- Also runs a Linux VM with Container run-time
- Desktop Application
- Kubernetes single node-cluster
- Docker Compose
- Start here: <https://docs.docker.com/docker-for-mac/>

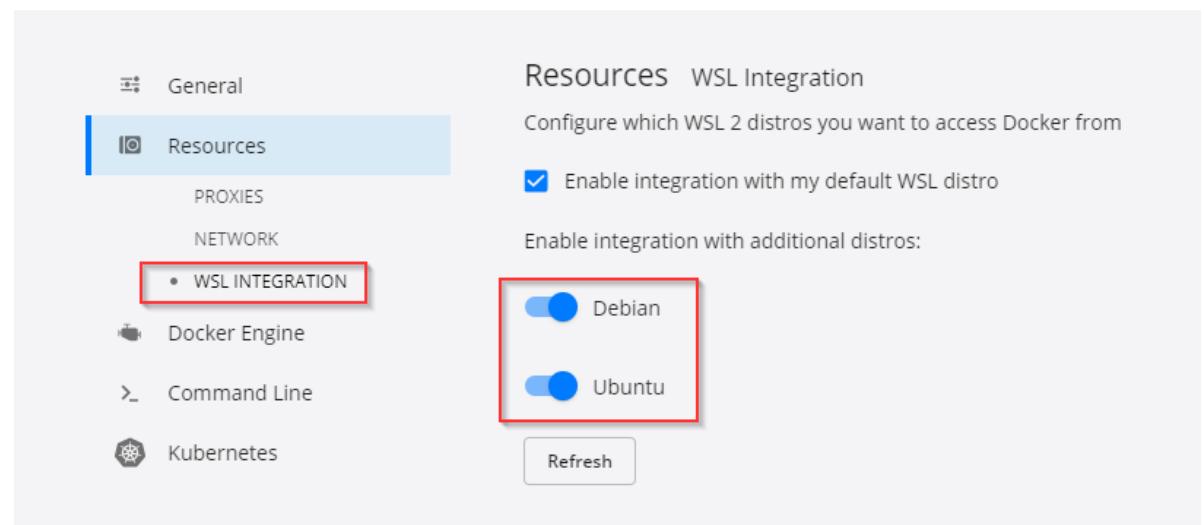
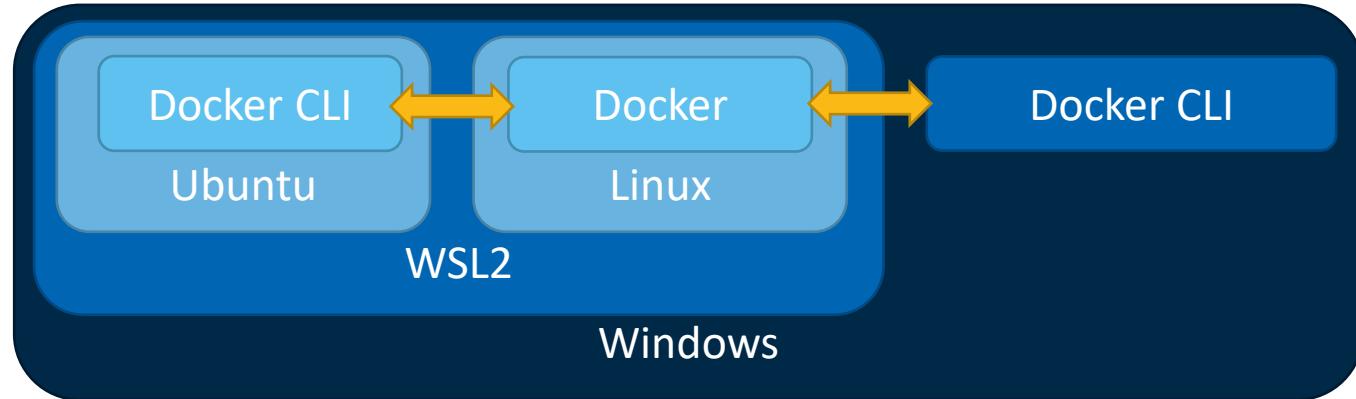


Docker.app



# Level 1 – Docker on Windows with WSL 2

- Uses native WSL2 Linux containers for Container run-time
- Full integration in guest Linux like Ubuntu running in separate WSL container each
- Built and run-time environment
- Kubernetes single node-cluster
- Docker Compose
- Not free any more for corporate users
  - Free for small companies and open source projects
- Still the best to use desktop solution
- Start here:  
<https://docs.docker.com/docker-for-windows/wsl/>



# Level 1 – Rancher Desktop



- Free
- Platforms supported
  - Windows WSL
  - Linux
  - Mac Intel + Apple Silicon
- Supports Docker with **docker** cmd
- Or Containerd with **nerdctl** cmd
- Full WSL integration on Windows
- A build and run-time environment

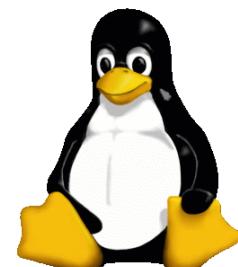
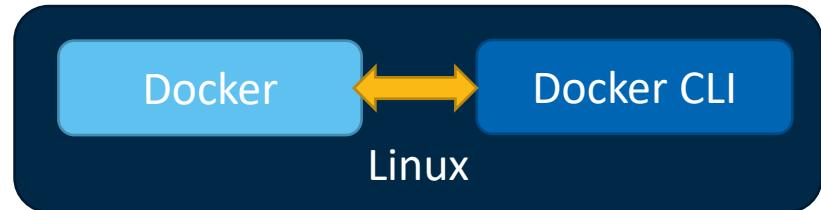
The screenshot shows the Rancher Desktop application interface. On the left, there's a sidebar with tabs: General, Port Forwarding, Images (which is selected), Troubleshooting, and Diagnostics. The main area is titled 'Images' and shows a list of images with columns for Image, Tag, Image ID, and Size. There are two entries: 'hclcom/domino' with tag '12.0.2' and 'Image ID' '3a53e043d80e', and 'hclcom/domino' with tag 'latest' and 'Image ID' '3a53e043d80e'. Below the main window is a 'Rancher Desktop - Preferences' dialog. This dialog has tabs for Application, WSL (selected), Container Engine, and Kubernetes. Under 'Container Engine', 'containererd' is selected (indicated by a blue dot). The 'General' tab is active in the preferences dialog. At the bottom right of the slide, there's a small '22' and the 'HCLSoftware' logo.

Image	Tag	Image ID	Size
hclcom/domino	12.0.2	3a53e043d80e	1.7 GiB
hclcom/domino	latest	3a53e043d80e	1.7 GiB

This screenshot shows the 'Rancher Desktop - Preferences' dialog. The 'WSL' tab is selected. It contains a checkbox for 'openSUSE-Leap-15.3' which is checked. The 'Container Engine' tab is also selected and shows two options: 'containererd' (selected) and 'dockerd (moby)'. The 'General' tab is active. The 'Application' tab lists 'Expose Rancher Desktop's Kubernetes configuration and Docker socket to Windows Subsystem for Linux (WSL) distros'. The 'Kubernetes' tab is visible at the bottom.

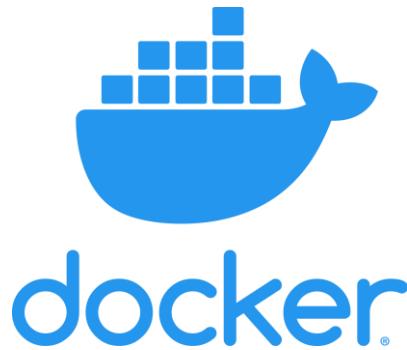
# Level 1 – Docker on Linux

- CentOS/RedHat, Debian, Fedora, Ubuntu  
(separate install instructions per platform)
  - Docker service controlling **containerd** + Docker Client
  - Runs native on Linux
  - Okay for smaller production deployments
  - You can optionally install Docker Compose
  - No Kubernetes included
  - Build and run-time environment
- 
- Start here: <https://docs.docker.com/engine/install/>

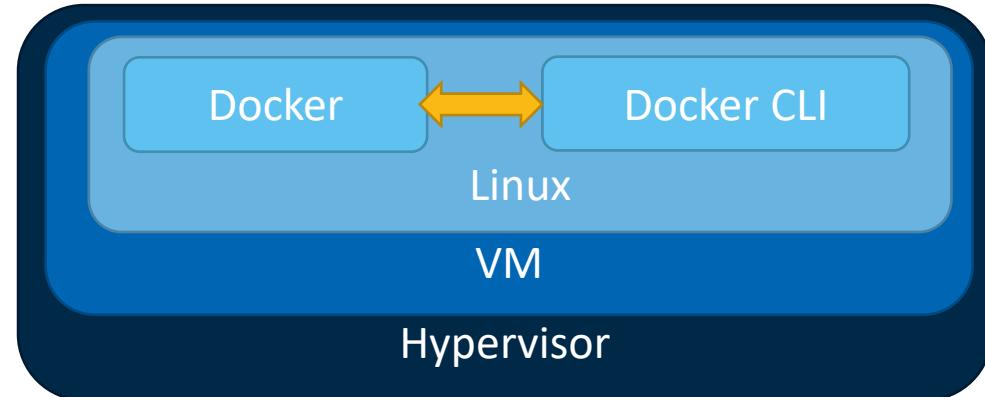


Level 1 conclusion : Win? MacOS? Linux? It's always Linux in one way or the other

# Level 2 – Docker Server



- Intended for small deployments and POC type environments
- Easiest to install and configure
- Build and run-time environment
- Does not support automatic scaling
- **Fully supported for production**



- System Recommendation
  - min. 2 CPU Cores,
  - 8 GB RAM,
  - 50 GB storage
- Linux Operating System
  - e.g. CentOS Stream 8+ or RedHat 8+
- Docker CE 20+

# Level 3 – Kubernetes (k8s)

- <https://kubernetes.io/>
  - Aka “vanilla” kubernetes
  - Built for enterprise deployments at scale
  - Contains management and orchestration
  - Open source & free of charge
  - Runs on virtualized machine or on HW
- 
- Various “flavors” and branded versions of Kubernetes exist
    - e.g. <https://k3s.io/> a lightweight distribution for IoT



**kubernetes**



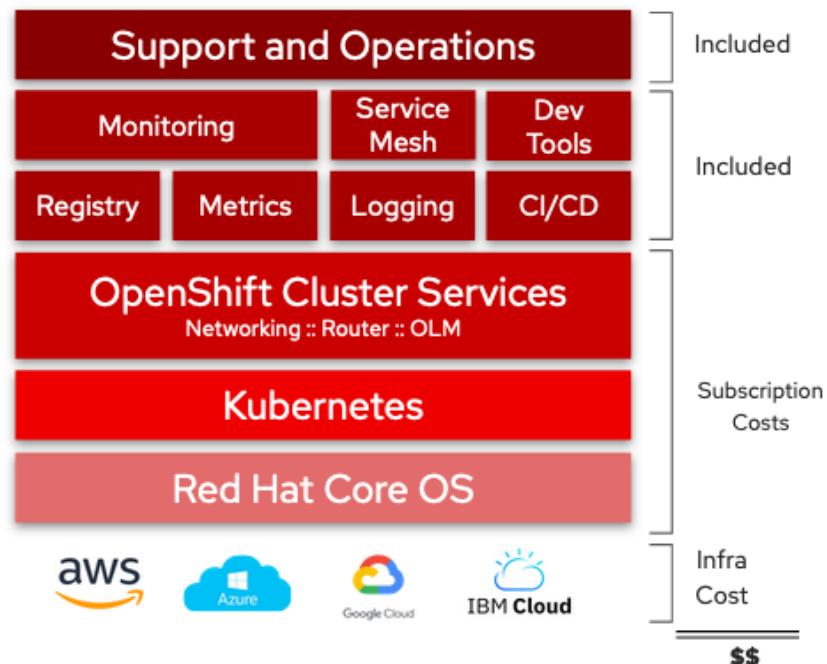
# Level 3 – OpenShift



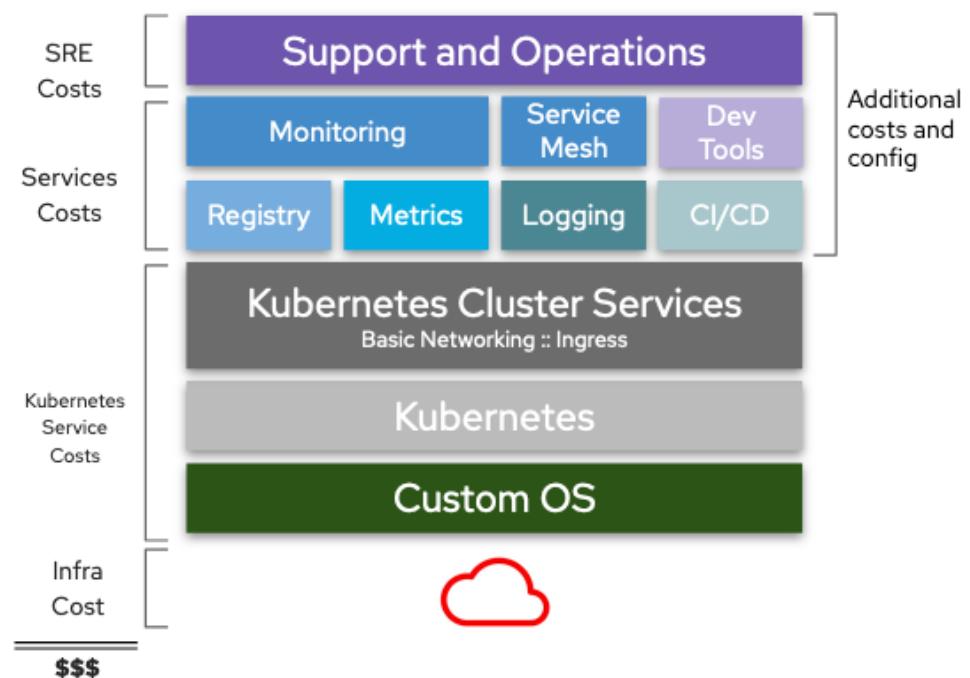
<https://www.openshift.com/>

- It's a flavor of Kubernetes
- Built for enterprise deployments at scale
- Contains management and orchestration
- Strong focus on security

## Red Hat OpenShift Managed Services



## Other Kubernetes Services

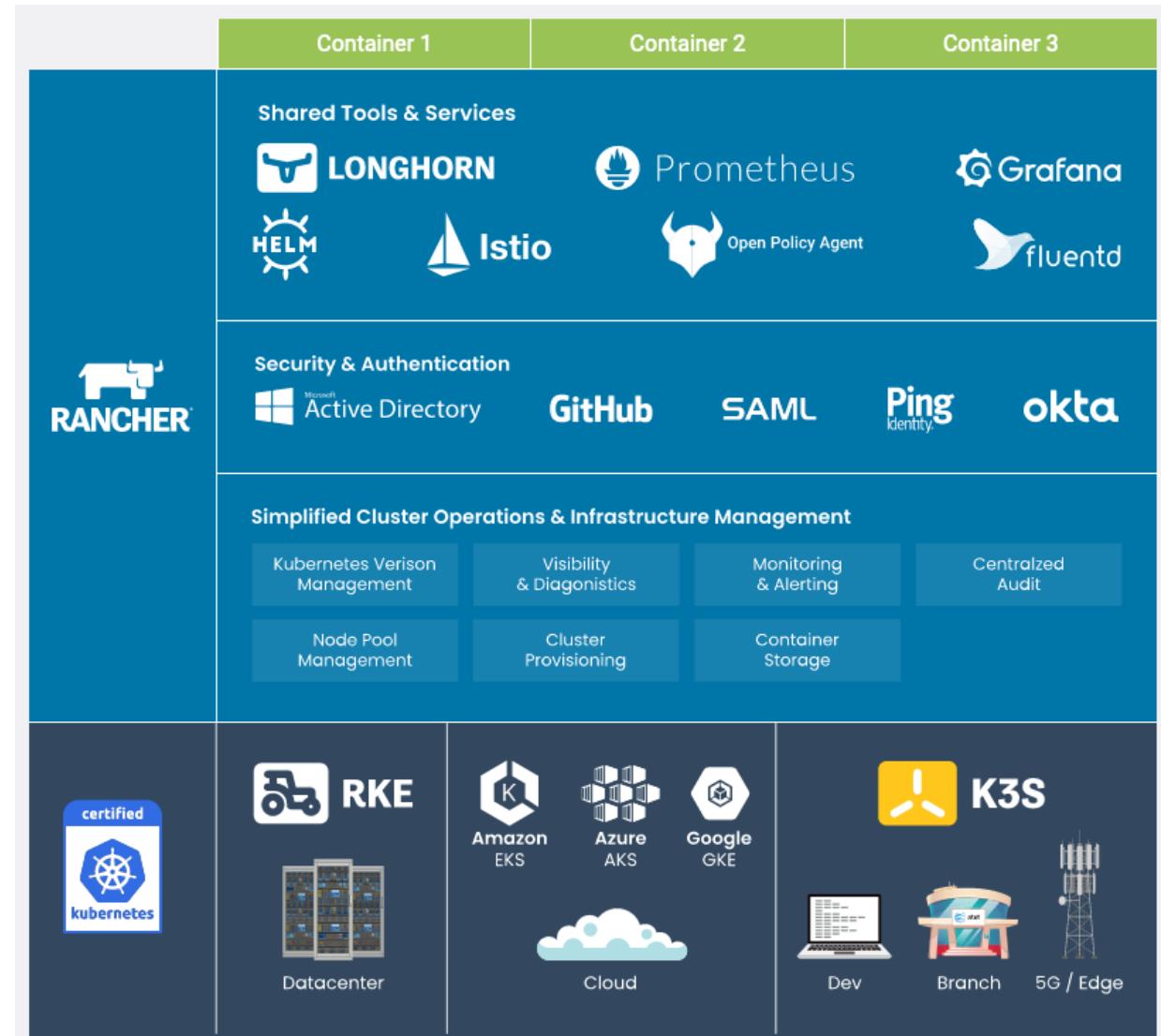


# Level 3 – Rancher

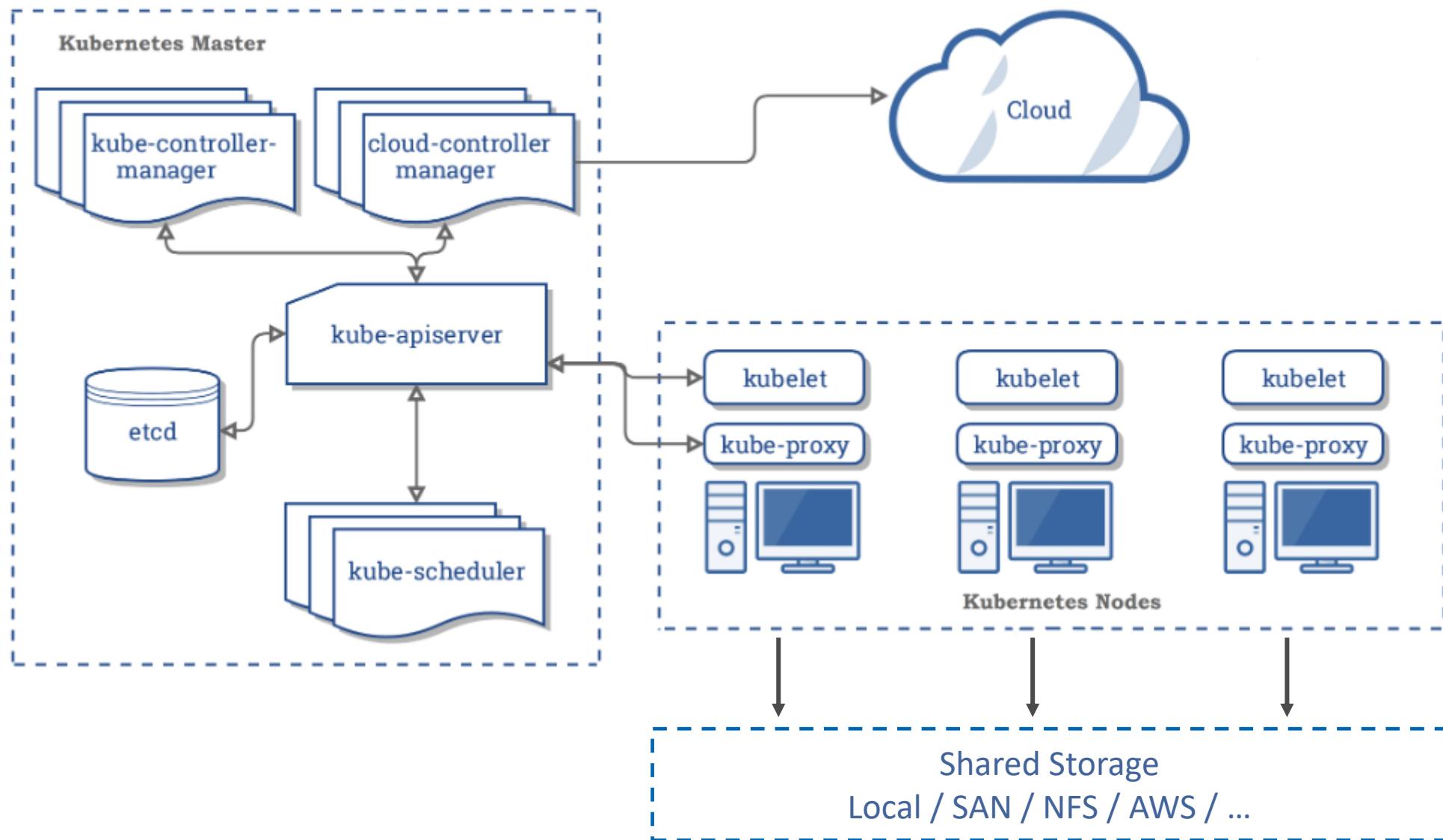
- <https://rancher.com>
- It's a flavor of Kubernetes
- Built for enterprise deployments at scale
- Contains management and orchestration
- Runs on virtualized machine or bare metal



**RKE 2**



# Kubernetes Architecture: Don't worry if this looks complicated!



# Container Terminology for Domino Admins

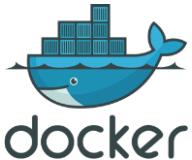


Image	=	Template
Container	=	Application
Volume	=	Domino Data Directory
CLI	=	Domino console
Registry	=	OpenNTF.org

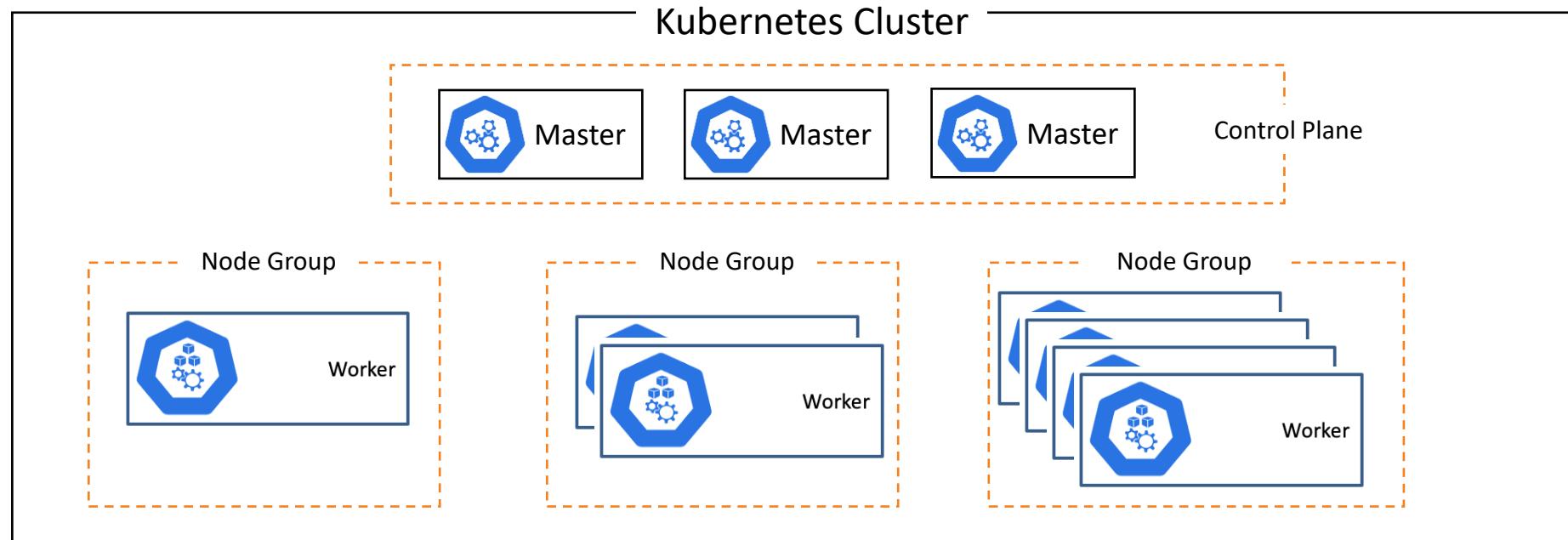


Node	=	Domino Server
Pod	=	Application
kubelet	=	Admin4.nsf
etcd	=	Names.nsf Clbdbdir.nsf
Controller	=	DDM Probe
Kube-proxy	=	Passthru
Service	=	Domino Cluster
YAML file	=	Your perfect Documentation



# Kubernetes

- Kubernetes = Containers Orchestration
  - High Availability
  - Failover
  - Auto scale
  - Workload Management



# Basic Kubernetes Concepts



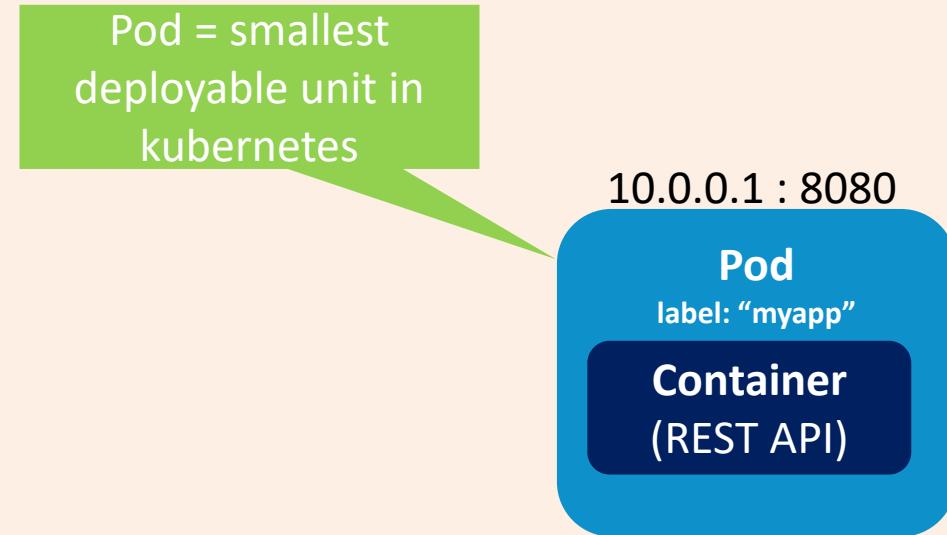
- **Pod**
  - Smallest scheduled component containing one or more containers exposing **ports**
- **Service**
  - Maps to one or more **Pods** providing an application service
  - Load balances requests among pods mapped for this Service
  - Can be externally exposed on a **NodePort / Cluster IP**
  - Or consumed via an **Ingress**
- **Ingress**
  - Provides external access via HTTP(S)

## Kubernetes Cluster

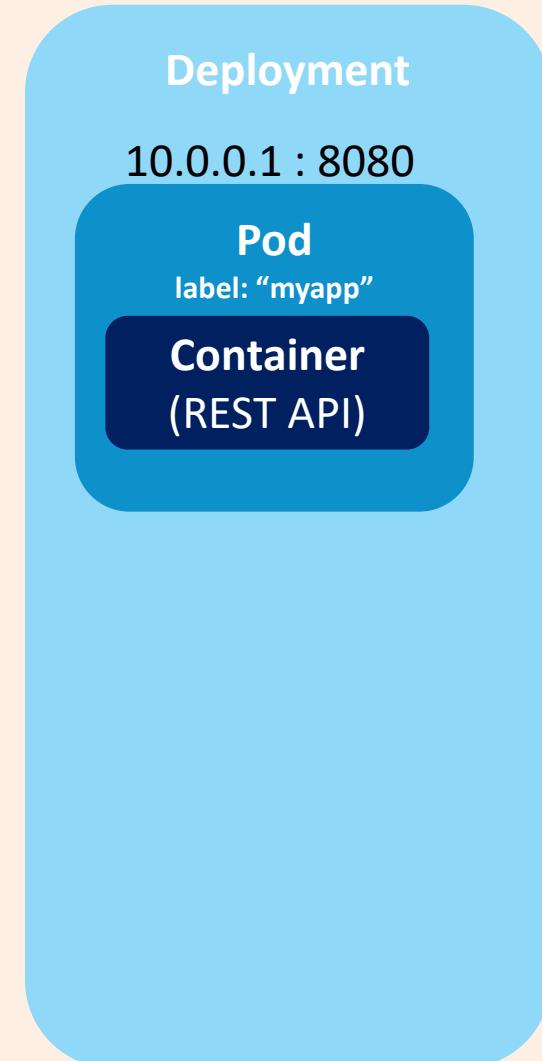
## Kubernetes Cluster

Container  
(REST API)

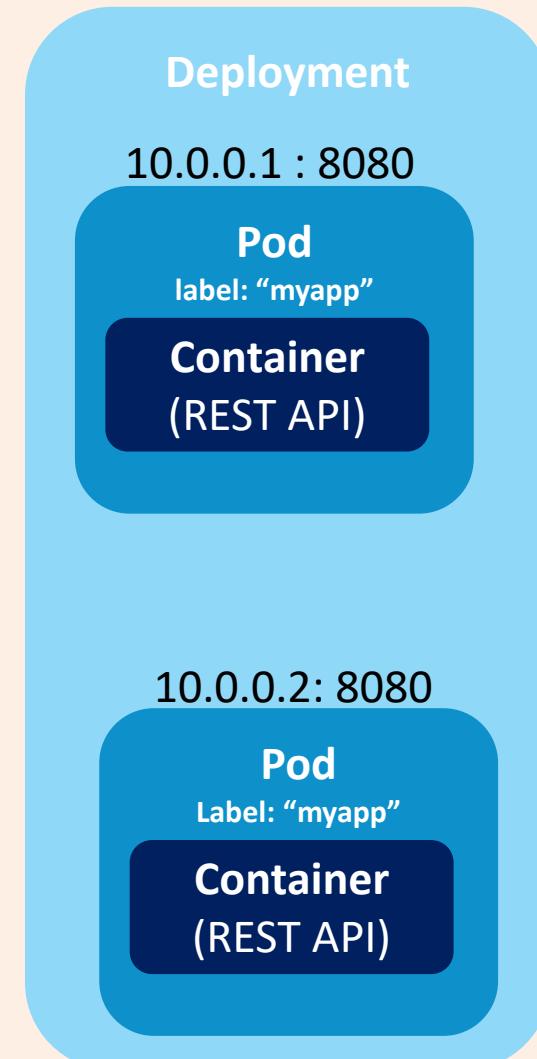
## Kubernetes Cluster



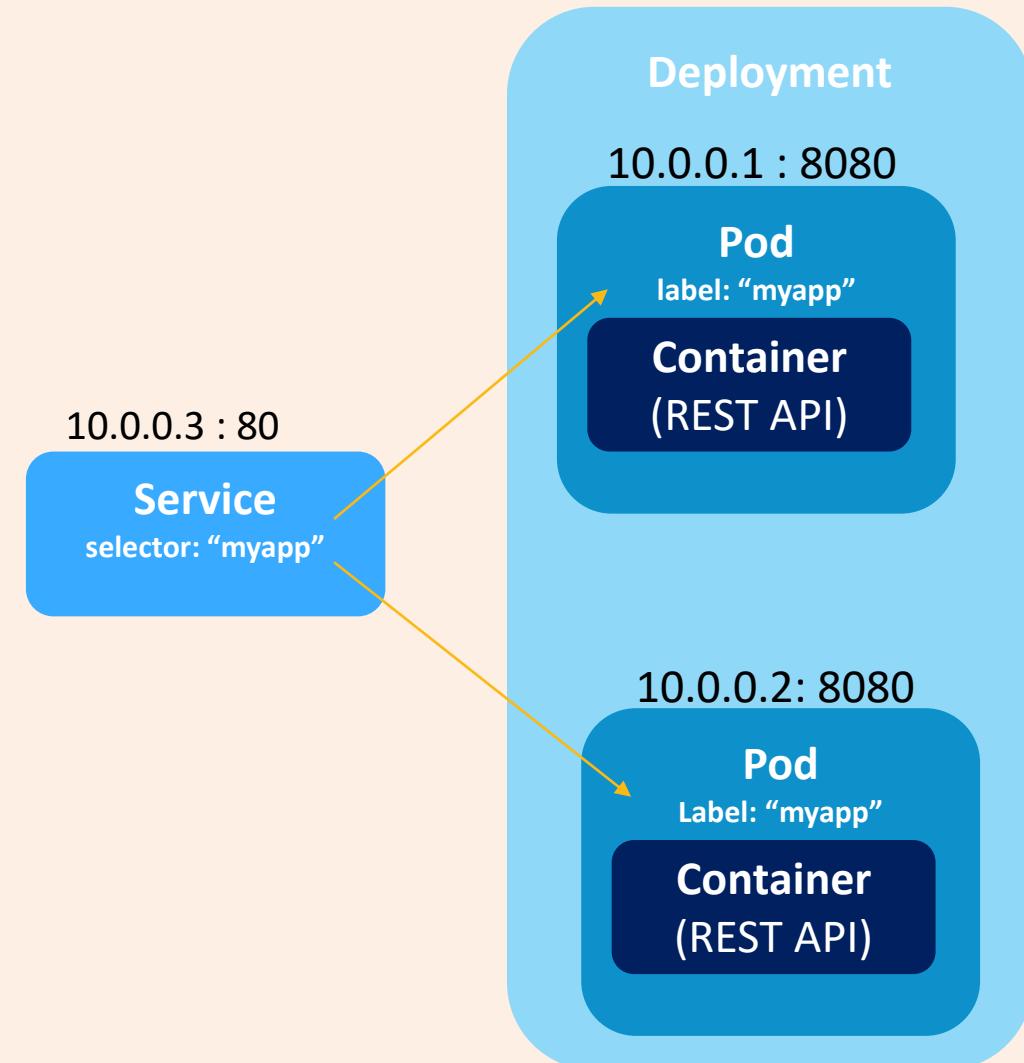
## Kubernetes Cluster



## Kubernetes Cluster

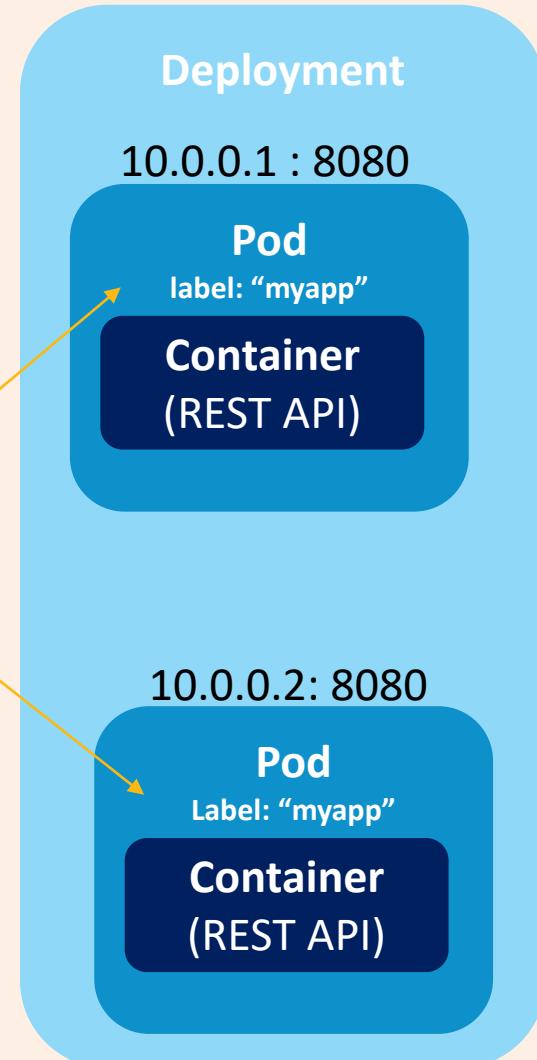
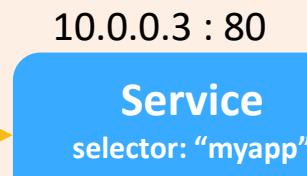


## Kubernetes Cluster



## Kubernetes Cluster

35.15.67.219 : 443



## Kubernetes Cluster

35.15.67.219 : 443



10.0.0.3 : 80



10.0.0.4 : 80



Deployment

10.0.0.1 : 8080



10.0.0.2: 8080



# Basic Kubernetes Concepts



- **Deployment / ReplicaSet / DaemonSet**
  - Defines Pods to be created for applications, which are stateless and can scale
- **StatefulSet**
  - Defines Pods for a stateful application, which usually needs one or more volumes
- **Persistent Volume Claim (PVC)**
  - Provides storage for stateful applications mapped to **Containers**
- **ConfigMap**
  - Configuration mounted into a container (similar to a volume)
- **Secret**
  - Information like TLS credentials, passwords etc, mounted into a container (similar to a ConfigMap..)

# Kubernetes Storage

## Persistent Volume

Allocated to the entire Kubernetes Cluster

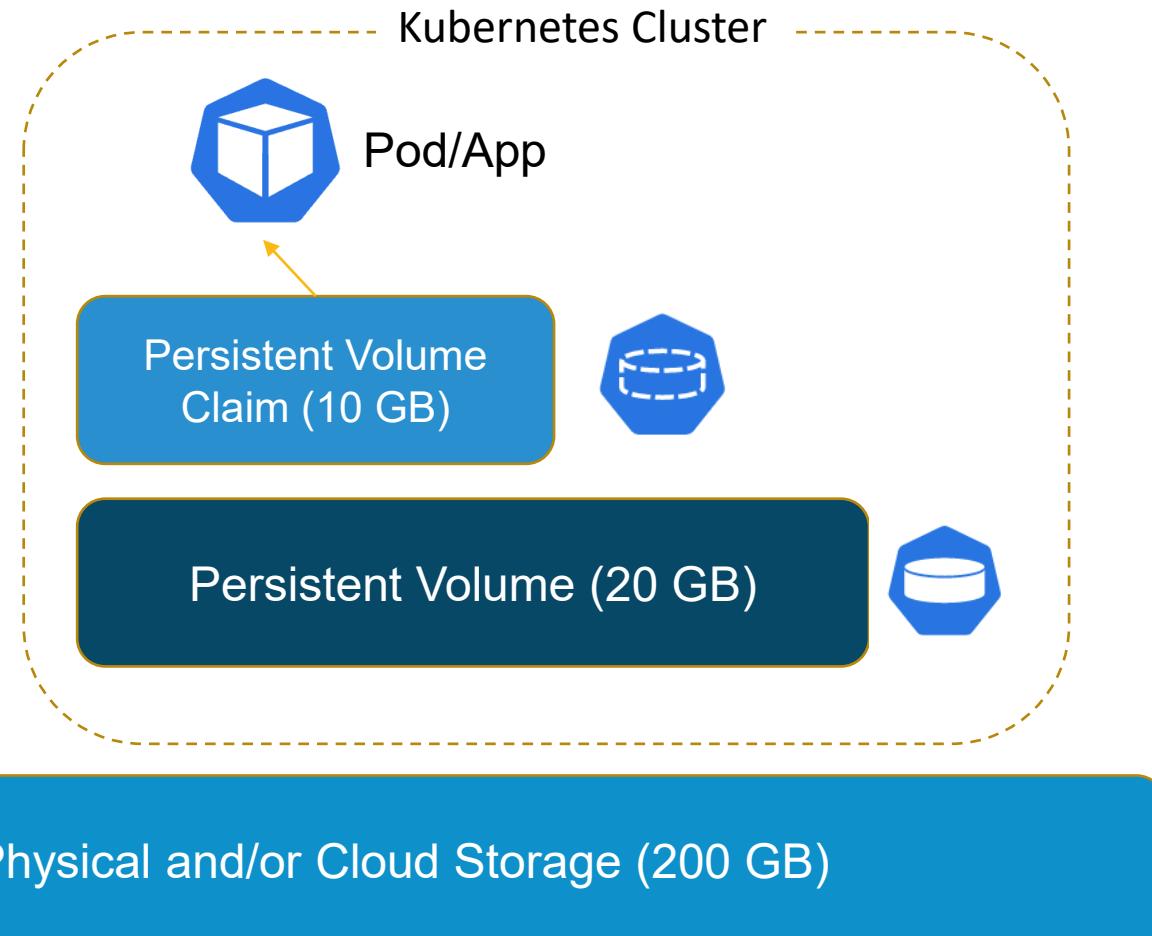


## Persistent Volume Claim

Storage needed (claimed) by a Pod

## StorageClass

Automates the process to create PV dynamically as Pods asks for PVC



# Container Storage Interface driver (CSI)



- “CSI” is the standard for implementing storage drivers
  - Drivers can implement additional services like
    - Volume expansions
    - Snapshots
- Cloud providers have their own CSI driver for their storage
  - Example: Hetzner Cloud Server
    - <https://github.com/hetznercloud/csi-driver>
- Rancher Longhorn enterprise storage uses iSCSI to provide storage in a cluster
  - Supports volume expansion, replication, snapshots

The screenshot shows a GitHub repository page for the 'hetznercloud/csi-driver' project. The repository has two files: 'skaffold.yaml' and 'README.md'. The 'README.md' file is the primary content displayed:

**Container Storage Interface driver for Hetzner Cloud**

This is a [Container Storage Interface](#) driver for Hetzner Cloud enabling you to use ReadWriteOnce Volumes within Kubernetes & other Container Orchestrators. Please note that this driver **requires Kubernetes 1.19 or newer**.

**Getting Started**

Depending on your Container Orchestrator you need to follow different steps to get started with the Hetzner Cloud csi-driver. You can also find other docs relevant to that Container Orchestrator behind the link:

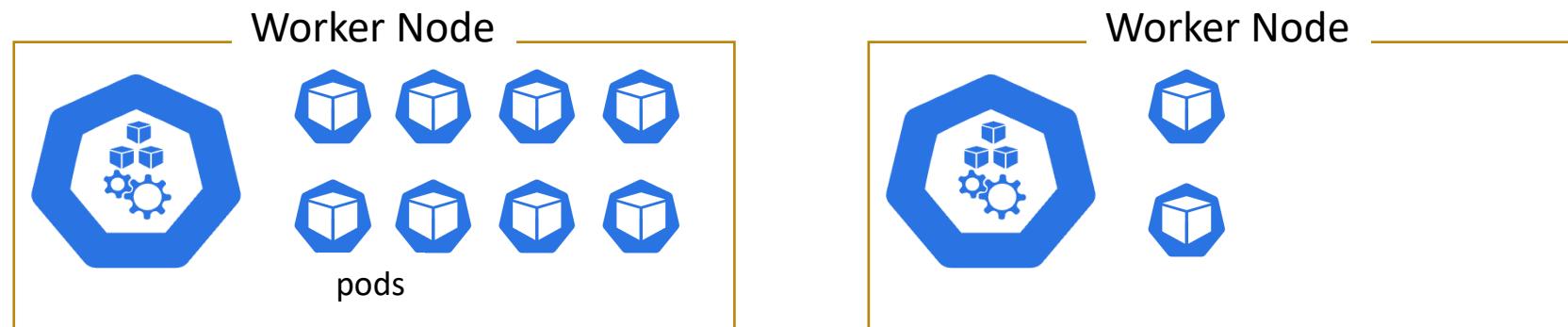
# Container Terminology for Domino Admins

			
Image	=	Template	
Container	=	Application	
Volume	=	Domino Data Directory	
CLI	=	Domino console	
Registry	=	OpenNTF.org	
Namespace	=	Domain	
Node	=	Domino Server	
Pod	=	Application	+ DDM
kubelet	=	Admin4.nsf	
etcd	=	Names.nsf Clbdbdir.nsf	
Controller	=	DDM Probe	
Kube-proxy	=	Passthru	
Service	=	Domino Cluster	
YAML file	=	Your perfect Documentation	



# Kubernetes Nodes and Pods

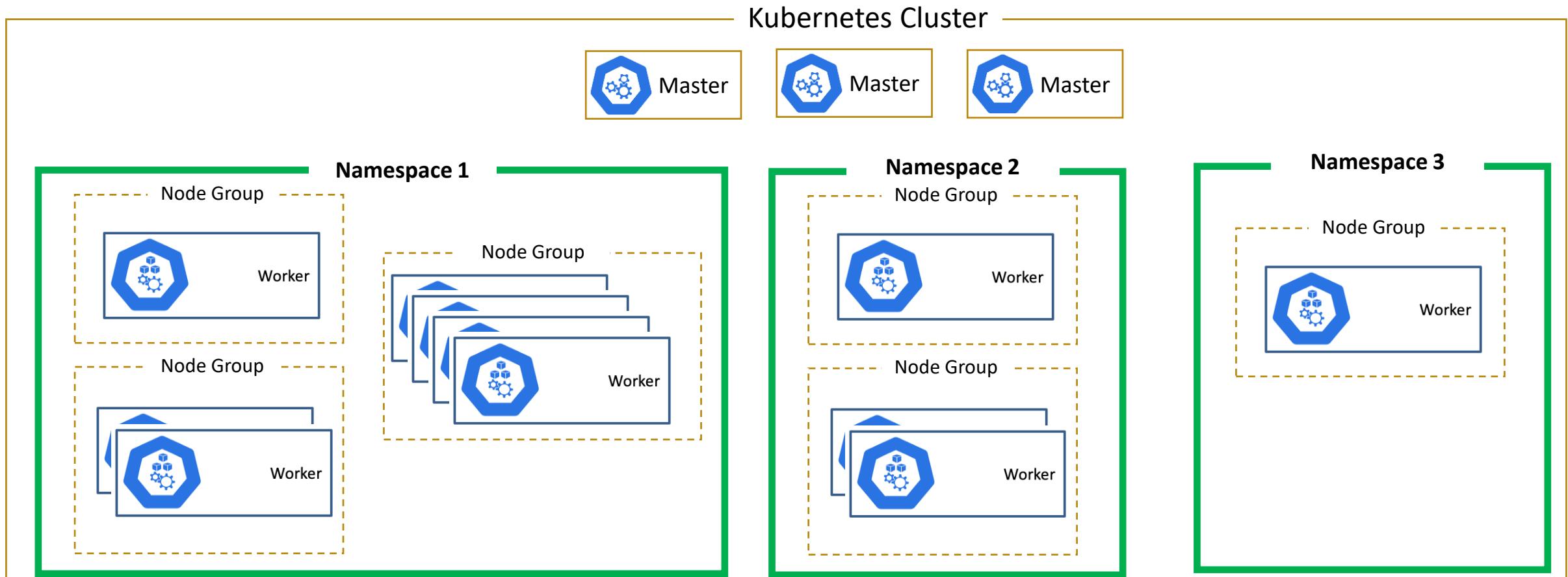
- You can use a single cluster for many workloads or applications (Namespaces)
- One application can run on one or multiple nodes
- Containers run inside pods
- When Applications need to scale, more Pods are added.
- When Nodes run out of compute power, new Nodes are added.





# Kubernetes and Namespaces

- Allows to organize different workloads (or Applications) in the same cluster





# Which provider?

Managed Kubernetes or self-hosted?

## Managed

Easier to start.

The provider manages the cluster.

Provider offers related services  
(Load Balancers, Storage, ...)

## Self-hosted

Test environment – easy.  
Production environment – complex.

You manage the cluster.

You manage networking and storage.





# Image registry

A catalog of private/public images

**IMPORTANT:** HCL Domino is not free software. If you build a container image for a Domino server, you cannot publish it on an open image registry.

You must use a private image registry protected from public access.

You also have to provide access credentials to the registry (as a part of custom config settings).

---

Registries offered by cloud providers



AWS Elastic  
Container  
Registry



Google  
Artifact  
Registry



Azure  
Container  
Registry

Registries you can deploy yourself



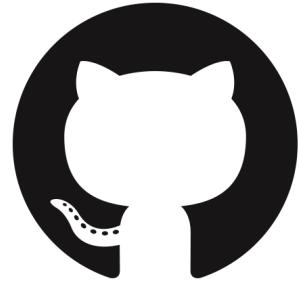
**Red Hat**  
**Quay**

# HCLSoftware

Tools & Technologies  
You should know



# Git & GitHub



- **GIT**

- De-facto standard in software development today
- Originally invented by Linus Torvalds to coordinate Linux kernel development
- Command line & GUI tools

- **GitHub**

- Many open source projects are mainly available and developed on GitHub
- HCL Software provides many open source projects under the main umbrella
  - <https://github.com/HCL-TECH-SOFTWARE>
  - CertMgr, Domino 12 Backup and Domino container image provide repositories

# Git



- A tool you should know today!
- Many software developers and companies use Git to manage their source code
- For our workshop you will need some simple git commands
  - “**git clone**” – to clone a project
  - “**git checkout develop**” – to switch to the develop branch of the project
  - “**git pull**” – to update your local repo

---

- TIP: Git client on Windows provides **Linux tools**
  - For example **Idd.exe** is very convenient for analyzing Windows dependencies
  - There is even **vi** - Very useful if you log into a machine via SSH

# Git uses Mark Down standard (.md)



- Very simple and reduced syntax to write easy to use documents
  - <https://en.wikipedia.org/wiki/Markdown>
  - <https://www.markdownguide.org/>
- HCL Documentation project C-API Docs
  - <https://github.com/HCL-TECH-SOFTWARE/domino-c-api-docs>
- GitHub Pages & MKDocs
  - <https://opensource.hcltechsw.com/domino-c-api-docs/>

The screenshot shows a website with a blue header bar containing the text "HCL Domino C API Documentation". Below the header, there's a navigation menu with a "Home" link. The main content area contains several links: "Overview of the HCL C API for Notes/Domino", "Building Windows Applications", "Building UNIX Applications", "Domino\_Database", "Security", and "Access Control Lists". At the bottom of the page, there's a footer with copyright information: "Copyright © 2023, HCL Technologies Limited - Change cookie settings" and "Made with Material for MkDocs".

Home

Overview of the HCL C API for Notes/Domino

Building Windows Applications

Building UNIX Applications

Domino\_Database

Security

Access Control Lists

Refer to the [HowTo](#) documents

Copyright © 2023, HCL Technologies Limited - Change cookie settings  
Made with Material for MkDocs

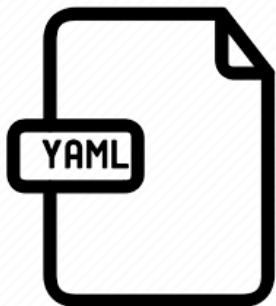
# {JSON} – A standard you should know



- <https://www.json.org>
- Most configuration files today are in JSON
- It replaces XML in many cases.
- If you never looked into JSON you have start now!
- The most popular parser on Linux and Windows:
- **JQ** – <https://stedolan.github.io/jq/>
- Very powerful and included in the Linux distributions



# YAML – Another standard you should know



- <https://yaml.org/>
- YAML / YML is often used in the container world to describe a configuration
  - For example used in docker-compose and K8s configuration
- Very reduced & simplified format
- What their homepage says:
  - “YAML: YAML Ain't Markup Language”
  - “What It Is: YAML is a human friendly data serialization standard for all programming languages.”
- Tool for Linux: <https://mikefarah.gitbook.io/yq/>

# XML → JSON → YAML

XML	JSON	YAML
<pre>&lt;Servers&gt;   &lt;Server&gt;     &lt;name&gt;Server1&lt;/name&gt;     &lt;owner&gt;John&lt;/owner&gt;     &lt;created&gt;123456&lt;/created&gt;     &lt;status&gt;active&lt;/status&gt;   &lt;/Server&gt; &lt;/Servers&gt;</pre>	<pre>{   Servers: [     {       name: Server1,       owner: John,       created: 123456,       status: active     }   ] }</pre>	<pre>Servers:   - name: Server1     owner: John     created: 123456     status: active</pre>

- Source: <https://developer.ibm.com/technologies/containers/tutorials/yaml-basics-and-usage-in-kubernetes/>
- Great video with all you need to start: <https://www.youtube.com/watch?v=1uFVr15xDGg&t=116s>

# CURL – A tool you should know



- De-facto standard for HTTP/HTTPS and other internet protocol operations
- **A - Command line tool** shipped with Windows and Linux
- **B. LibCurl** is an open source SDK used by applications (usually dynamically linked → “DLL”)
  - Domino and CertMgr leverages **LibCurl** internally as well
  - Free PDF book: <https://everything.curl.dev/>
  - Founder and lead developer Daniel Stenberg / <https://daniel.haxx.se/>
  - Commercial support available for developers by WolfSSL
- We will use the command line version on Linux and Windows to download files
  - Example: **curl -L https://download.com/nshcertool -o /usr/bin/nshcertool**
- Tip: Download current curl version to replace old version included in Windows
  - <https://curl.se/windows/>



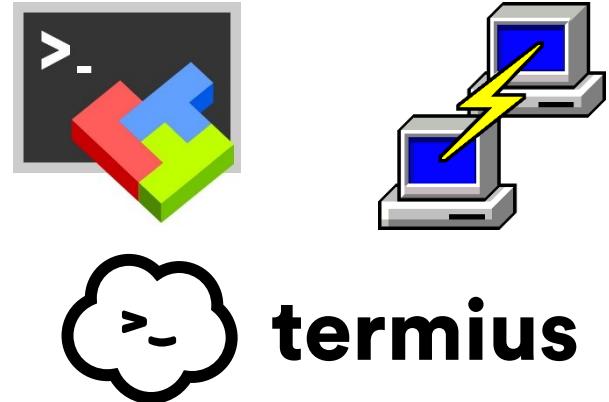
- Initiated for #DACHNUG49
  - <https://dnug.de/dnug-lab/>
- Stable, permanent lab environment covering the latest Domino, Traveler, Nomad, ST, .. features
- Can be used by all members for
  - Checking out new functionality
  - Collaborate on projects
- Reference implementation for Domino on different platforms
  - Windows, Linux, Docker, Podman
  - **New:** Kubernetes environment
    - RKE2 + Rancher + Longhorn storage



# Secure Shell SSH



- Standard and most secure connection to servers
  - User/Password
  - **SSH key (recommended --> Disable password authentication!!)**
  - TOTP
  - Or a combination of all three?
- Global configuration + authorized keys per user
- Used for
  - Terminal sessions
  - Secure copy (scp, sftp)
  - Execution of remote commands
  - Tunnel connections
- Recommended tools
  - MobaXterm
  - Putty
  - Terminus
- OpenSSH server for Windows → <https://github.com/PowerShell/Win32-OpenSSH/releases>



# HCLSoftware

## Domino 12 One Touch Setup

“Auto Config”



# Domino 12 One-Touch



- HCL Documentation
  - [https://help.hcltechsw.com/domino/12.0.2/admin/inst\\_onetouch.html](https://help.hcltechsw.com/domino/12.0.2/admin/inst_onetouch.html)

The screenshot shows a web browser displaying the HCL Domino documentation. The title of the page is "One-touch Domino setup". The left sidebar contains a navigation menu with the following items:

- Domino® server setup program
  - > Using the Domino® server setup program
  - One-touch Domino setup**
    - > Preparing input parameters for one-touch Domino setup
    - > Invoking one-touch Domino setup
  - Troubleshooting one-touch Domino setup
  - Understanding the Domino® server certification log
  - > Domino® server registration

The main content area starts with a brief introduction: "Use one-touch Domino setup to simplify setting up a server." It then explains the new feature in Domino 12: "In previous versions of HCL Domino®, setting up a Domino server involved multiple steps. Starting with Domino 12, you can use one-touch Domino setup to set up a server in a single step. You invoke one-touch Domino setup by referring to a JSON file or a set of environment variables that contain the setup configuration information." Below this, it describes the steps required: "The steps you take to use one-touch Domino setup depend on whether you provide input through a JSON file or system environment variables and the Domino platform that you use." A section titled "Using one-touch Domino setup you can:" lists four bullet points: "Set up servers", "Set up an ID vault", "Create and update applications and documents and enable and run agents. This feature is available only through JSON file input.", and "Register users and create user mail files. This feature is available only through JSON file input.". At the bottom, it states: "One-touch Domino setup is supported on Domino on Docker, Windows, and UNIX platforms."

# Domino 12 One-Touch (OTS)

- Two different configuration options
- **Simple ENVIRONMENT variable setup**
  - Very simple to use
- **JSON based setup**
  - More complex, but very powerful and flexible
  - Including Application configuration (update documents, create databases, documents, etc.)
  - Schema validation for JSON format
- OTS automation is fully integrated into the HCL Community Domino and HCL Domino image
  - Provides automated ways to download JSON file
  - Replaces {{ ENVIRONMENT\_.. }} placeholders



# HCLSoftware

Domino 12  
on Docker & Podman



podman

# Podman vs Docker



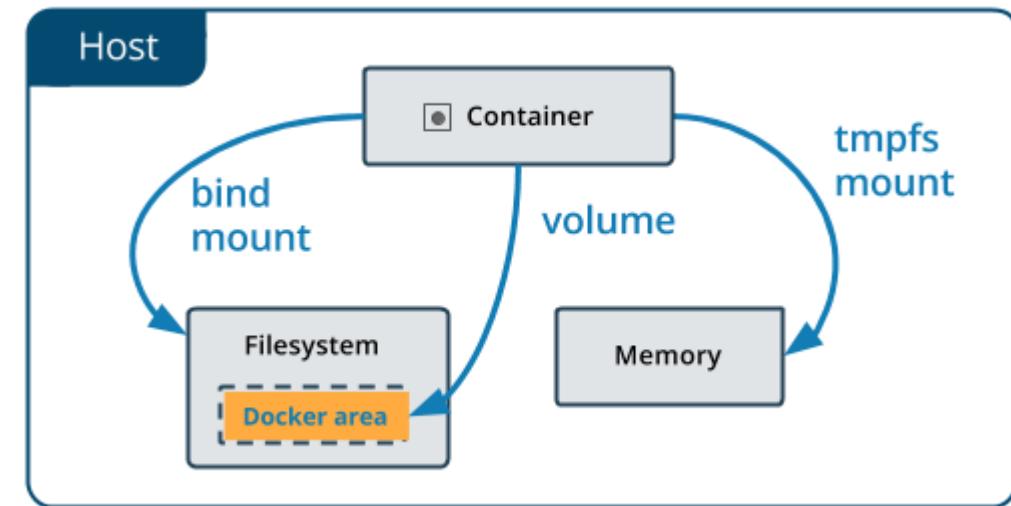
podman

- Podman is another **run-time** and **build** platform shipping with Redhat and SUSE distributions
- Installation is a single command:
  - “**`yum install -y podman`**” (Redhat/CentOS – and clones)
  - “**`zypper install -y podman`**” (SUSE)
- In contrast to Docker, Podman does not use a **daemon**
  - it is not running as **systemd** service and all containers are started directly by **podman**
- 1:1 replacement for Docker functionality
  - With some minor differences in detail
- **podman-compose** is **not** a 1:1 replacement for **docker-compose**
  - But you can use docker-compose also with podman
    - See <https://www.redhat.com/sysadmin/podman-docker-compose> for details

# Run your first Domino Server in a Container

```
• docker run -it -d --name domino12 \
--hostname=marvel.domino-lab.net \
-p 80:80 -p 443:443 -p 1352:1352 \
-v vol-notesdata:/local/notesdata \
--cap-add=SYS_PTRACE \
--env-file env_domino \
```

**domino-container:v1202\_09152022prod**



<b>-it -d</b>	Run in interactive mode but detached
<b>--name</b>	Defines name of container
<b>--hostname</b>	Defines host name inside the container
<b>-p host-port:container-port</b>	Publishes and maps a port outside the container
<b>-v volume-name:/local/notesdata</b>	Maps a volume to the container
<b>--cap-add=SYS_PTRACE</b>	Allows GNU debugger to attach to processes (NSD)
<b>--env-file</b>	Specifies environment variable file

# Interact with your Domino Server



- Daily administration should be performed via Admin client
- Admins can also start a **shell** into the running container if really needed
- **docker exec -it domino12 /bin/bash**
  - Return via “exit”
  - Use “-u 0” option for root access
- “**docker logs domino12**”  
to see the logs from startup & configuration operations



# Domino Start Script Commands

- Invoked inside the container via “**domino**” command

```
show server

Lotus Domino (r) Server (Release 5.0.3 (Intl) for UNIX) 06/11/2022 08:27:32 PM

Server name: lego-builder/FactoryTour
Server directory: /local/notesdata
Partition: .local.notesdata
Elapsed time: 11 days 04:21:24
Transactions/minute: Last minute: 0; Last hour: 0; Peak: 0
Peak # of sessions: 2 at 04/11/2022 03:53:41 AM
Transactions: 0
Availability Index: 100 (state: AVAILABLE)
Message Tracking: Not Enabled
Shared mail: Not Enabled
Number of Mailboxes: 1
Pending mail: 0 Dead mail: 0
Waiting Tasks: 0
Transactional Logging: Enabled
> close
Live Console closed.
```

<b>domino console</b>	Runs Domino live console
<b>domino help</b>	Lists commands with short description
<b>domino start/stop</b>	Starts/Stops Domino server inside the container
<b>domino archivelog</b>	Archives notes.log file and compresses it
<b>domino nsd</b>	Runs a manual NSD
<b>domino res</b>	Shows Domino processes, memory and MQs

# dominoctl – Domino Container Control Script

- Optional admin script designed to run the Domino Community + new HCL image
  - Simplifies Domino container administration
  - Supports **docker & podman** (+ **nerdctl** on Rancher Desktop)

- Run “**dominoctl**” from a prompt
  - **dominoctl** provides command-line help
  - Provides start/stop, update, config & troubleshooting commands

- **Installation**

- Clone or download the GitHub project <https://github.com/nashcom/domino-startscript>
  - Example: `git clone https://github.com/nashcom/domino-startscript.git`
- Run install script
  - `./domino-startscript/install_dominoctl`



# Domino Container Run-Time Support

- The new image is tested on

- **Domino Docker 20.10+**
  - **Podman 3.3.0+**



**podman**

- Working but not actively tested

- **Kubernetes (K8s), Redhat OpenShift**



- **Docker Desktop on Windows/Mac, Rancher Desktop**



**K3S**



- Customers cannot expect HCL Support to help with container run-time platform issues

- **Container itself is supported** similar to Linux distributions meeting the basic system requirements

# Import Docker Image

- Import image to your Docker host via “docker” command remains the same
- **docker load --input Domino\_12.0.2\_Container\_Image.tgz**

```
aae852b00c54: Loading layer [=====>] 44.54kB/44.54kB
6da388c94871: Loading layer [=====>] 239.1kB/239.1kB
92487d3c7076: Loading layer [=====>] 56.1MB/56.1MB
dbe8a9ec8c94: Loading layer [=====>] 1.077GB/1.077GB
Loaded image: domino-container:v1202_09152022prod
```

- Verify the Docker image is on your Docker server

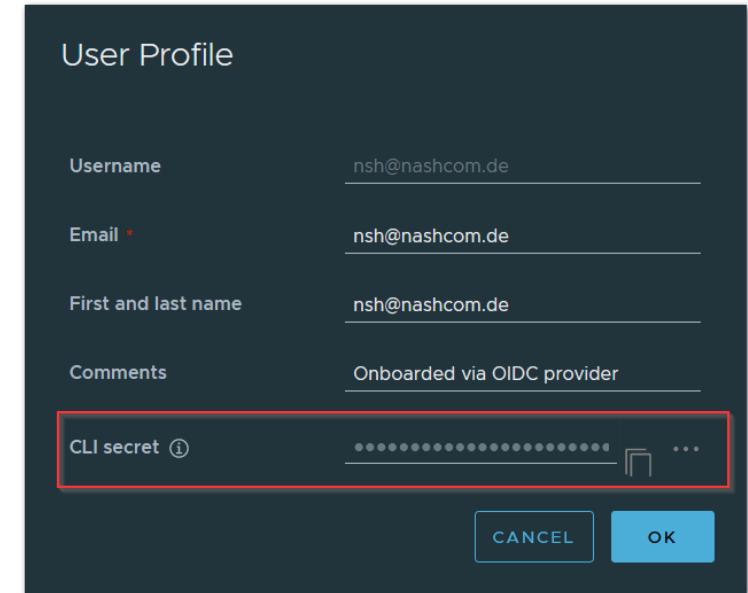
**docker images**

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
domino-container	v1202_09152022prod	c3919b6d975c	3 weeks ago	1.34GB

# Pull Image from HCL Harbor Registry



- Domino isn't available on the Docker public Registry
  - But every customer on maintenance should have access to the HCL Harbor registry
- 1. Login with your Flexnet account to the Harbor registry
  - Also uses Okta SSO
- → <https://hclcr.io/>
- 2. Get an authentication token
- 3. Login with your e-mail address + token
  - `docker login hclcr.io`
- 4. Pull image
  - `docker pull hclcr.io/domino/domino-container:v1202_11032022prod`



# Domino 12 “One Touch Setup”

- Domino 12 introduced “One Touch Setup”
  - Provides **automated setup + server configuration**
  - Cross platform, integrated into core -- Not only available for containers!
    - Very useful in the container world
- Two modes
  - a.) **Environment variables** – Designed to be used with containers
    - Basic functionality
  - b.) **JSON File** → Allows to configure many additional settings!
    - Create databases, documents, configure ID-Vault, TLS Credentials ..

# Domino V12 “One Touch Setup”

- For basic configuration use environment variables
  - More advanced functionality is available passing a JSON file (e.g. via volume mount)
  - Environment on Docker host referenced in docker run statement via `--env-file env_domino`
- `env_domino`

```
SetupAutoConfigure=1
SERVERSETUP_SERVER_TYPE=first
SERVERSETUP_ADMIN_FIRSTNAME=John
SERVERSETUP_ADMIN_LASTNAME=Doe
SERVERSETUP_ADMIN_PASSWORD=domino4ever
SERVERSETUP_ADMIN_IDFILEPATH=admin.id
SERVERSETUP_ORG_CERTIFIERPASSWORD=domino4ever
SERVERSETUP_SERVER_DOMAINNAME=DominoDemo
SERVERSETUP_ORG_ORGNAME=Domino-Demo
SERVERSETUP_SERVER_NAME=domino-demo-v12
SERVERSETUP_NETWORK_HOSTNAME=domino.acme.com
```

# Log Files and config files inside the Container

- **/tmp/domino-container/install\_domino.log**
  - Domino installation log (from the build process → just for information)
- **/tmp/domino-container/data\_update.log**
  - Log for extracting the data directory from **/domino-container/install\_data\_domino.taz**
- **/domino-container/domino\_ver.txt → /local/notesdata/domino\_ver.txt**
  - Version of the Domino image used for triggering updates
- **/local/notesdata/notes.log**
  - Domino server log output file
- **/etc/sysconfig/rc\_domino\_config**
  - Start Script configuration

# Stop a Domino Server cleanly!

- Docker by default sends a **SIGTERM** signal to the main process of the container
  - If the main process does not stop within **10 seconds**, a **SIGKILL** signal is send to the container!
  - This would not allow Domino to shutdown cleanly
  - Always specify `set --stop-timeout=90` on the run statement
- Specify a higher wait time if needed for specific situations in the shutdown command
  - `docker stop --time=120 domino12`

# Working with Docker

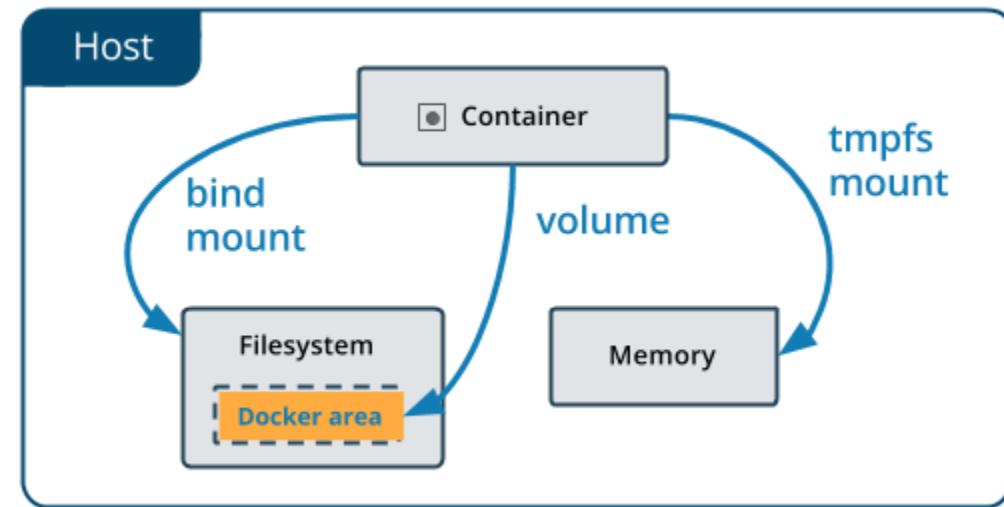


<b>docker images</b>	Lists locally available images
<b>docker run</b>	Runs a <u>new</u> container based on an image
<b>docker start/stop</b>	Starts/stops an <u>existing</u> container
<b>docker ps [-a]</b>	Shows all running process – All processes using –a
<b>docker inspect</b>	Shows detailed information for a container or image
<b>docker exec</b>	Executes a command or shell <u>inside</u> the container
<b>docker system prune</b>	Removes all stopped containers, unused networks, dangling images (images not tagged)

# Docker Volumes



- By default all data is stored in the container
- For applications with local storage requirements, this does not work well
- Docker supports “volumes” which are mapped into the container
  - The data from the local directory will be copied to the volume at first run when the volume is empty
  - The default implementation is a local disk
    - You can either create a volume manually, mount existing directories or let Docker create it
- <https://docs.docker.com/storage/volumes/>



# Docker Volumes



- You can specify an existing volume in your run statement
  - `docker run --rm -it -v /local/data1:/local/data centos:latest bash`
- Or let Docker create a local volume
  - `docker run --rm -it -v test-data1:/local/data centos:latest bash`
  - Default location: `/var/lib/docker/volumes`
- Or for example use a NFS mount on a NAS
  - `docker volume create --driver local --opt type=nfs --opt o=addr=192.168.96.41,rw --opt device=/data/docker_vol --name nfsvol`
  - `docker run --rm -it -v nfsvol:/local centos:latest bash`

# Docker Volume Commands



**docker volume ls**

Lists all volumes

**docker volume inspect my-vol**

Shows details about one volume

**docker volume create my-vol**

Creates a new volume

**Docker volume rm my-vol**

Removes a volume!

**Docker system df**

Shows used and free space

**Docker volume prune**

**Use with care:** Removed all volumes not referenced by any container

# Docker Documentation



- Great documentation
  - If you google you are hitting the official website most of the time
- Reference for all commands
- Official website
  - <https://docs.docker.com>
- Includes installation instructions for all platforms

The screenshot shows the Docker documentation website's navigation bar at the top, which includes links for Home, Guides, Product manuals, Reference (which is underlined), and Samples. The main content area is titled "Reference documentation" and contains two tables: one for "File formats" and one for "Command-line interfaces (CLIs)".

**File formats**

File format	Description
<a href="#">Dockerfile</a>	Defines the contents and startup behavior of a single container
<a href="#">Compose file</a>	Defines a multi-container application

**Command-line interfaces (CLIs)**

CLI	Description
<a href="#">Docker CLI</a>	The main CLI for Docker, includes all <code>docker</code> commands
<a href="#">Compose CLI</a>	The CLI for Docker Compose, which allows you to build and run multi-container applications
<a href="#">Daemon CLI (<code>dockerd</code>)</a>	Persistent process that manages containers

# Docker Compose



- Uses a **YAML** file to define one or multiple containers
  - Multiple containers can join the same **network** also defined by the configuration
  - Sametime meetings is using docker-compose to bring up multiple containers as one **service**
- Many other projects use **docker-compose.yml** files to describe their services
- In the end it's still managing the same containers we already know in a different way
- Examples in the community image Git repository
- The docker-compose syntax is different than the standard to define pods in in Kubernetes
  - But the principles are similar
  - Podman has “**podman play**” bridging to the K8s world

# Docker Compose Commands



<b>docker-compose up</b>	Starts containers described in <b>docker-compose.yml</b>
<b>docker-compose down</b>	Stops the same containers and removes them
<b>docker-compose up -d</b>	Starts containers and detaches from console
<b>docker-compose logs</b>	Shows logs
<b>docker-compose ps</b>	Shows running containers
<b>docker-compose stop</b>	Stops containers without removing them
<b>docker-compose rm</b>	Removes containers

# Docker Compose Example



```
version: '3.6'
services:

  domino:
    image: hclcom/domino:latest
    container_name: domino-acme-01
    hostname: domino-acme-01.acme.com
    stop_grace_period: 60s

    environment:
      LANG: en_US.UTF-8

    cap_add:
      - SYS_PTRACE

    ports:
      - 1353:1352
      - 80:80
      - 443:443

    volumes:
      - domino_vol:/local

volumes:
  domino_vol:
    name: domino_acme_01_local
    external: false
```

- Shows all the information you already know
- Can bring up multiple containers together defining a service
  - Like the Sametime Meeting server
- Defines resources like volumes and networks separately
  - In this example you see a Domino data volume mounted at **/local**
- Can use variable names specified in an **.env** file

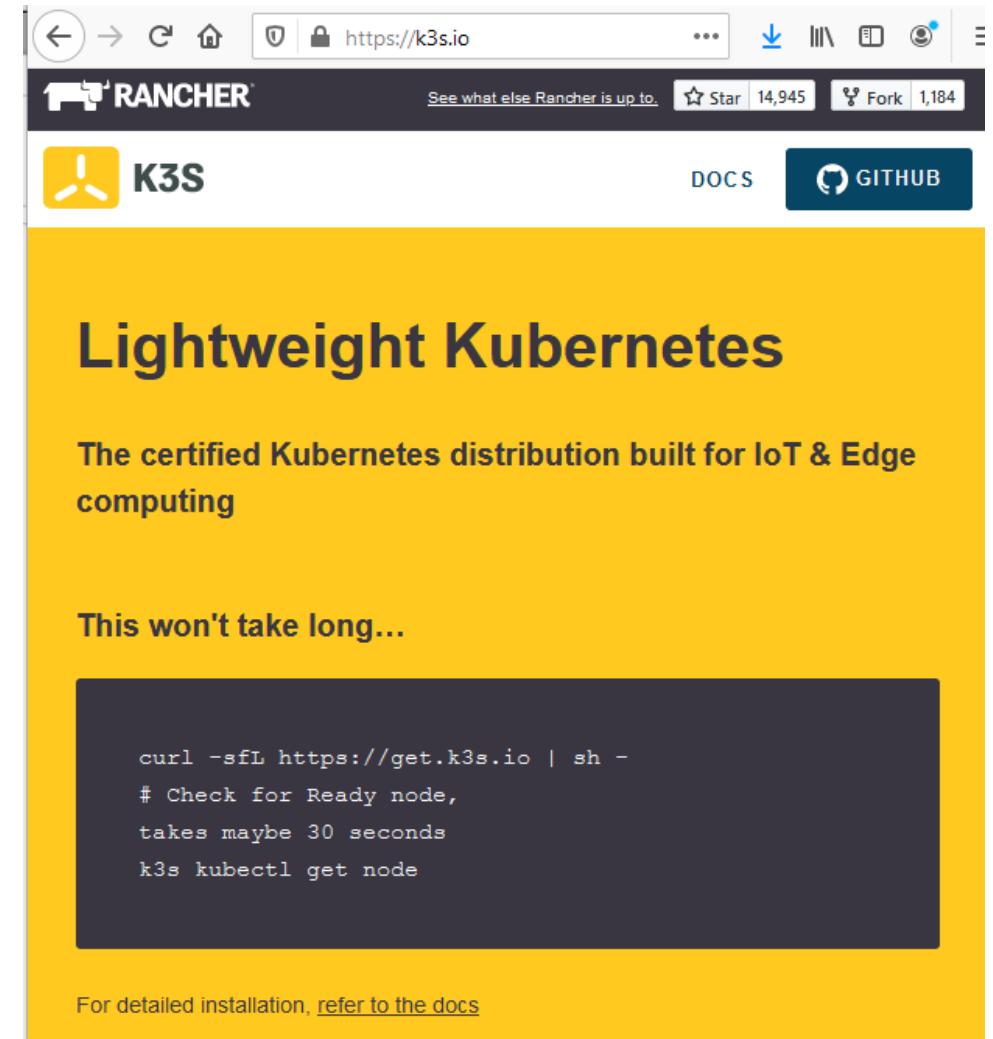
# HCLSoftware

## Domino 12 on Kubernetes



# Rancher K3s – Kubernetes in 1 Minute!

- Kubernetes & other distributions setup
- Can take a while and needs some resources!
- **K3s** works with **2 GB RAM** and **1 CPU** core
  - Very small production grade K8s!
- <https://k3s.io/>
- A single command installs the K3s
  - `curl -sfL https://get.k3s.io | sh -`
- Check installation
  - `kubectl get node`



# “Hello world” Kubernetes

- Create your first pod: `kubectl apply -f nginx.yml`
- Needs a pod definition in YAML

```
apiVersion: v1
kind: Pod

metadata
  name: hello-nginx
  labels:
    app: demo
spec:
  containers:
    - name: nginx
      image: nginx:latest
```



# Working with kubectl

- Kubectl is the management command-line tool for K8s
  - Similar to **docker-compose.yml** files are used to describe pods, services, storage, network, etc

<b>kubectl get node</b>	Shows infos about a K8s node
<b>kubectl apply -f domino.yml</b>	Apply a YML config with definitions for resources
<b>kubectl get all</b>	Overview of all resources defined
<b>kubectl describe pod/domino</b>	Show detailed information about a pod
<b>kubectl exec -it pod/domino -- bash</b>	Runs a bash into a pod
<b>kubectl logs pod/domino</b>	Shows logs for the pod
<b>kubectl delete -f domino.yml</b>	Delete resource defined in YML file

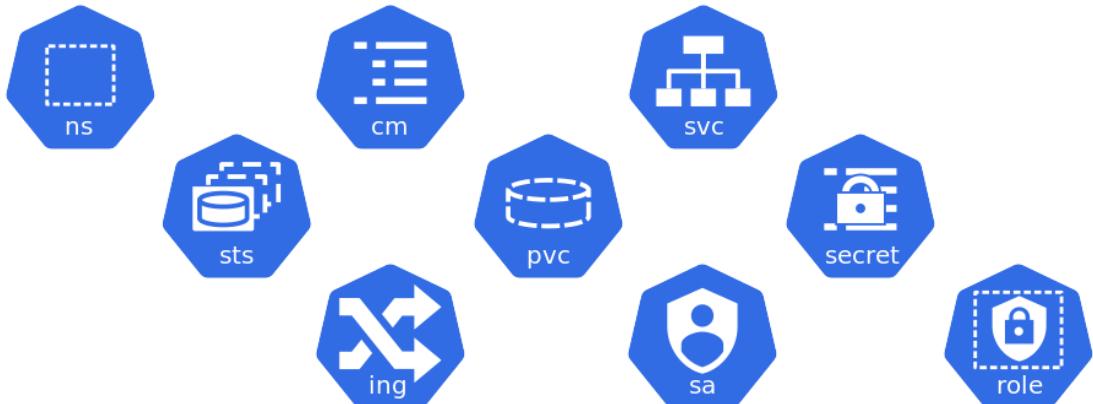


# Deployment script

All components that each Domino requires

## Helm Chart

Contains manifests that describe Kubernetes components needed to run an application.



## Values

A YAML file that contains values that can be parameterized.

You can use the default values or overwrite them with your custom values.

```
domino:  
  server:  
    name: Alpha  
    domainName: Space  
  network:  
    hostName: alpha.space.com  
  org:  
    orgName: Space  
    certifierPassword: password  
  admin:  
    firstName: Domino  
    lastName: Administrator  
    password: password
```



## Domino Helm Chart

An unofficial Helm chart for deploying Domino servers on Kubernetes cluster.



# Deployment script

## Domino Helm chart

main · 1 branch · 0 tags · Go to file · Add file · Code

File / Commit	Description	Time Ago
charts	Add serviceAccount support	3 weeks ago
docs	Publish new Helm chart for Domino and Do...	7 hours ago
examples	Add Example config files documentation	2 weeks ago
scripts	Polish scripts	yesterday
.gitignore	Add Example config files documentation	2 weeks ago
LICENSE	Initial commit	3 months ago
README.md	Create Deploy cert-manager script	2 months ago

README.md

### Helm Chart for HCL Domino

A Helm chart for HCL Domino server. Unofficial.

Helm charts  
domino  
domino-shared

Values description

Config examples

Helper scripts  
Kubernetes install  
cert-manager  
Ingress NGINX  
DNS records

<https://github.com/pkunc/domino-charts>



# Deployment script

## Domino Helm chart

Example: Deploying the first Domino server on Kubernetes.

```
helm install domino-shared pkunc/domino-shared \
  --namespace domino -f examples/domino-shared.yaml

helm install alpha pkunc/domino \
  --namespace domino -f examples/alpha-eks.yaml
```



# Download

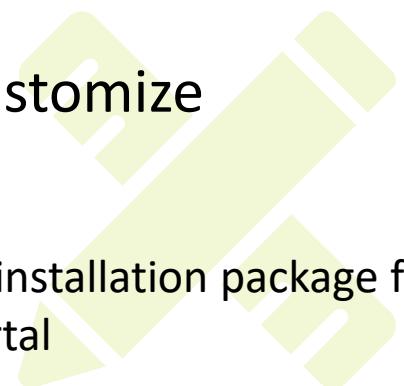
What do I need to download?

I want to build and customize

Download Domino on Linux installation package from  
FlexNet / new Download portal

Use the package to build a Domino image  
using a [Domino Community script](#).

Use parameters to build the image that fits  
your needs (version, add-ons)



OR

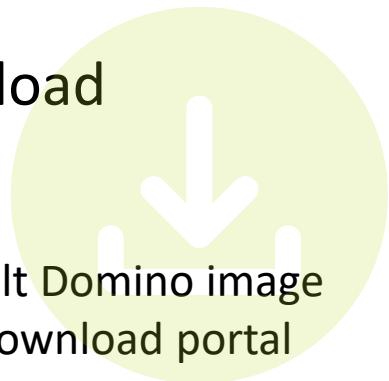


I just want to download

Download the already-built Domino image  
from FlexNet / new HCL download portal

or

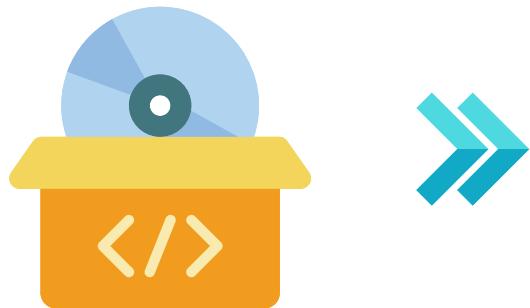
...pull the image from the HCL container  
registry [hclcr.io](https://hclcr.io)





# Upgrade

Upgrading the Domino server to a new version.



Build a new  
Domino image  
with a new tag  
and upload it to  
the registry.



Change the  
Domino version  
(image tag) in  
a Helm chart  
values file.



Apply Helm chart  
to upgrade the  
deployment.



The old pod is  
deleted. A new  
Domino pod with  
the new version is  
created.

The same applies when you want to modify add-ons (add Verse, Nomad, Leap, ...)



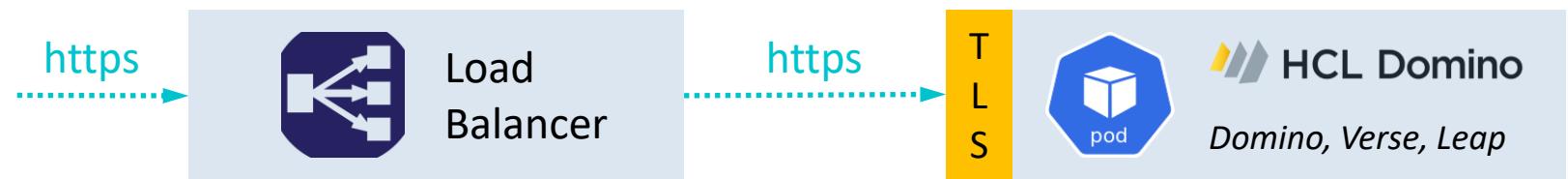
# Certificates

Who manages server TLS certificates?

A

Load Balancer

You have to manage TLS certificates on Domino.



B

Ingress

The [Ingress Controller](#) manages TLS certificates for you. You can automate it with *cert-manager*.



# HCLSoftware

[hcltechsw.com](http://hcltechsw.com)



# OS-level access

Access the operating system when the Domino runs

Many Kubernetes management and observation tools offer pod shell access.



```
[notes@alpha-domino-0 /]$ ls -l /local/notesdata/notes.ini  
-rw----- 1 notes notes 3196 Apr  2 14:19 /local/notesdata/notes.ini  
[notes@alpha-domino-0 /]$ |
```

```
kubectl exec -it alpha-domino-0 -n domino -- bash  
$ domino stop  
$ /opt/hcl/domino/bin/compact names.nsf  
$ domino start  
$ exit
```

You can also use the command line.



# Upload files

Upload files from your desktop to a pod

Sometimes you need to upload the file(s) from your workstation to a pod.

*Example:* Upload existing \*.nsf files.

Use the command line.

```
kubectl cp server-alpha.id domino/alpha-domino-0:/local/notesdata/server.id
```

local file

namespace

pod

remote file



For multiple files transfer, you can use my [script](#). It compresses a local folder, uploads it to a pod, and uncompresses the content in /local.

```
./upload-data.sh
```



# Download files

Download files from a pod to your desktop

Sometimes you need to download the file(s) from a pod to your workstation.

*Example:* Download cert, server, or admin Notes IDs.

Use the command line.

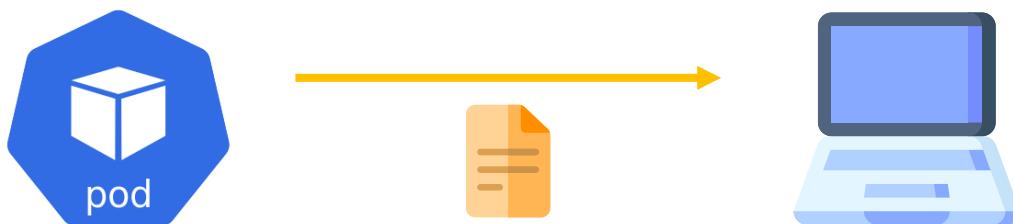
```
kubectl cp domino/alpha-domino-0:/local/notesdata/server.id server-alpha.id
```

namespace

pod

remote file

local file





## CertMgr & CertStore

Domain wide trusted root,  
private key & certificate management

# certstore.nsf – TLS Credentials

- **TLS Credential** = **private key + leaf certificate + chain (intermediates) + trusted root**
- Replaces “\*.kyr files”
  - Stored in **PEM** format (text with base64 encoded data)
- Can be created via
  - **ACME V2** protocol (Let's Encrypt & others)
  - Manual flows including import
  - Domino MicroCA (exportable in 12.0.2)
- Specify trusted roots used for client certificate verification
  - Used to be hidden in **kyr-file** and was difficult to manage

Submit Request Edit Document Cancel Examine Certificate(s)

## TLS Credentials

Main | Security/Keys | Manual | Comments |

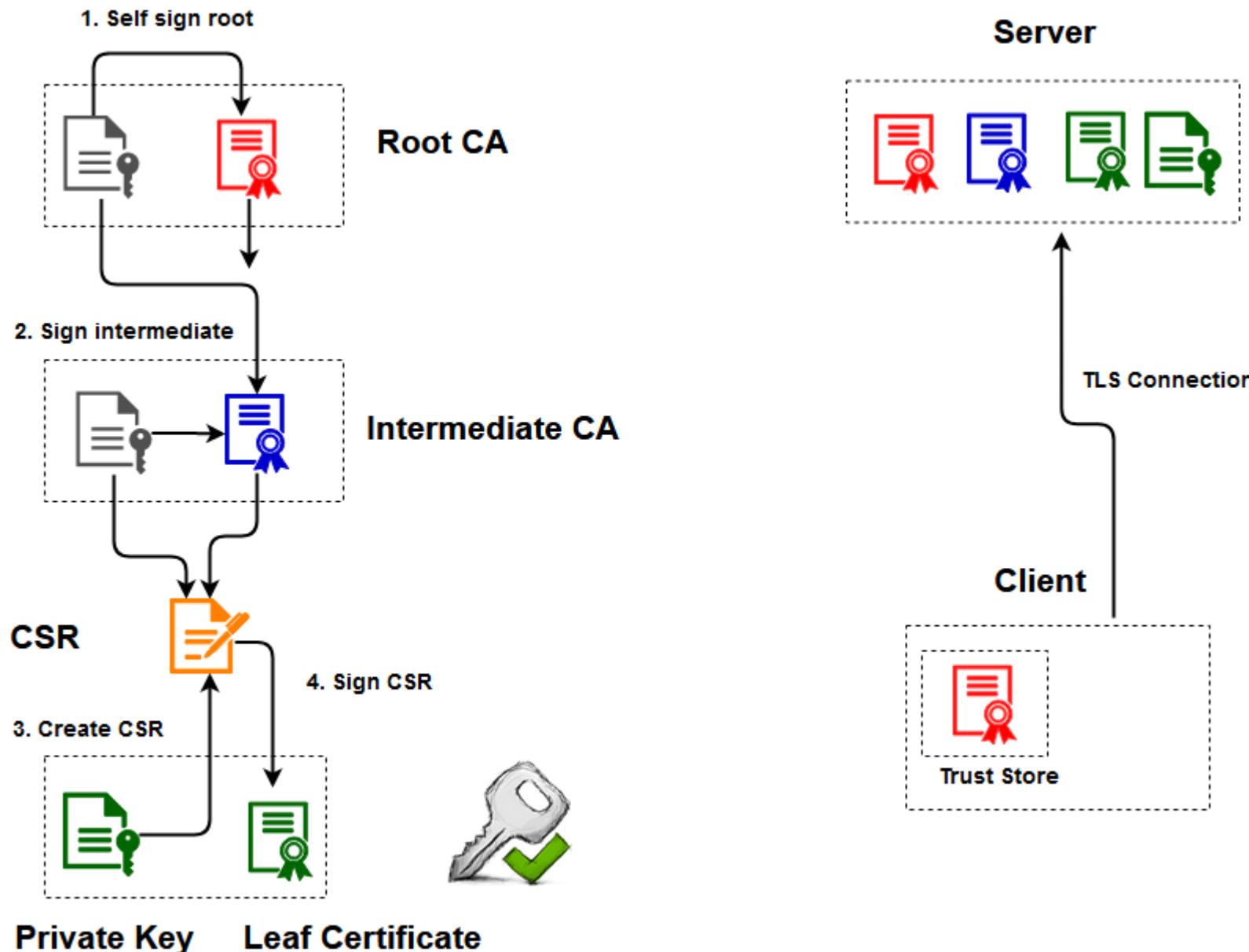
Main	
Status:	Issued
Host names:	pluto.csi-domino.com
Servers with access:	pluto/NotesLab
Status:	Valid
Certificate expiration:	Sun 05/30/2021 02:43:18 PM
Certificate renew date:	Fri 04/30/2021 02:43:18 PM
Certificate provider:	ACME
ACME account:	LetsEncryptProduction
Key type:	ECDSA
Curve name:	NIST P-384
Automatically renew:	30 days before expiration
Request key rollover:	
Keyring file:	

Trusted Roots Select Keywords

Keywords
<input type="checkbox"/> CN=Fake LE Root X1
<input checked="" type="checkbox"/> CN=ISRG Root X1/O=Internet Security Research Group/C=US
<input type="checkbox"/> CN=AAA Certificate Services/O=Comodo CA Limited/L=Salford/S
<input type="checkbox"/> CN=Buypass Class 2 Root CA/O=Buypass AS-983163327/C=NO
<input type="checkbox"/> CN=ISRG Root X2/O=Internet Security Research Group/C=US

PEM

# Keys, Certs, Chain & Root ..



- CA is a self signed certificate installed as a trust into a **local trust store**
- CA and intermediate/sub CAs have a private key signed by the next higher CA
- The last intermediate **“issuing CA”** signs the CSR resulting in a **leaf certificate**
- A **CSR** is signed by the private key with the information about what to certify (e.g. SAN, org)
- To verify a certificate the local root from the trust store + the full chain is used

# PEM Format

- Widely used de-facto standard to represent key and certificate information
  - [https://en.wikipedia.org/wiki/Privacy-Enhanced\\_Mail](https://en.wikipedia.org/wiki/Privacy-Enhanced_Mail)
- Text based format which can be merged and edited with any editor
  - **Base64** encoded form of the **binary DER format** + markers for begin and end of the data
- Supports unencrypted and encrypted keys
  - Encryption uses a “password/passphrase”
  - Data is still shown as text (base64 encoded)
- Many applications use separate PEM files for key and certificate chain
- A big benefit:
  - Update certs without re-encrypting the key

The diagram illustrates the PEM file structure. It features a central box labeled "PEM" containing two code snippets. The top snippet is titled "Exportable private key:" and shows a base64-encoded private key with BEGIN and END markers. The bottom snippet is titled "Certificate chain:" and shows a base64-encoded certificate chain with BEGIN and END markers.

```
Exportable private key:  
-----BEGIN ENCRYPTED PRIVATE KEY-----  
MIIBBkBiBgkqhkiG9w0BBQ0wVTA0BgkqhkiG9w0BBQwwJwQQEMfk6vWgAuIF+NSj  
uaWC9QICEAACASAwDAYIKoZIhvcNAgxFADAdBglghkgBZQMEASoEENsBxtotWQEy  
4nguFs01bCcEgaBgG1bFuszmeRQW41oDxuWzMBMap+OU0Gc44K1StcMUfeuhnDax  
LI1CtWMMIgvxIzD+0sSj74eG/CnNDox1L0jKyR8TePzAymQ+jd98x70FB1UPZwNC  
YtrL9+yth8j5EYMbCZz0E/Yux9BdLREKNA3idQZPuozqkK00q/lv2Urda5VanZyW  
qLa7tU1BB5hbrMOUhU2JmxVsyiSlmI/6J/Ix  
-----END ENCRYPTED PRIVATE KEY-----  
  
Certificate chain:  
-----BEGIN CERTIFICATE-----  
MIIBkTCCAtegAwIBAgIQcr4pwloz4qjYKEu323bLqTAKBggqhkJOPQQDAjAfMRAW  
DgYDVQQKDAd0YXNoQ29tMQswCQYDVQQDDAJDQTAeFw0yMTEyMDYxNjI2MT1aFw0Y  
MjEyMDgxNjI2MT1aMCsxEDAOBgNVBAoMB05hc2hDb20xFzAVBgNVBAMMDnd3dy5u  
YXNoY29tLmR1MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEfOIGtztQHC1JB9oo  
7fuB811yuAFn0Grj6dYIXLHEQWIPP273E8XX+m8IWySpP501Et80hWw0eSGvmuRJ  
HiEn56NjMEcwMAYIDVR0RBCkwJ4IOd3d3Lm5hc2hjb20uZGWCFXd3dy54bi0tYmNo  
ZXIta3ZhLmNvbTATBgnVHSUEDDAKBggRbgEFBQcDATAKBggqhkJOPQQDAgNIADBF  
AiEAvgusuRQojEXd8V00eTIYHeoWINiVzvx17vnLfQNPxY5cCIFDzE0n3xZE2+J6y  
spEapub3+REXxBdLNU8sGA8ZziQ4  
-----END CERTIFICATE-----
```

# Domino 12 One-Touch Setup Certificate Support

- Simple settings assuming defaults
  - More settings available in ENV and JSON config
- **Micro CA**
  - SERVERSETUP\_SECURITY\_TLSSETUP\_METHOD=**dominoMicroCA**
- **Import PEM**
  - SERVERSETUP\_SECURITY\_TLSSETUP\_METHOD=**import**
  - SERVERSETUP\_SECURITY\_TLSSETUP\_IMPORTFILEPATH=**my.pem**

```
"security": {  
    "ACL": {  
        "prohibitAnonymousAccess": true,  
        "addLocalDomainAdmins": true  
    },  
    "TLSSetup": {  
        "method": "import",  
        "retainImportFile": true,  
        "importFilePath": "wildcard_nashcom_org.pem",  
        "exportPassword": "Super42Secret007Password4Key"  
    }  
}  
  
"security": {  
    "ACL": {  
        "prohibitAnonymousAccess": true,  
        "addLocalDomainAdmins": true  
    },  
    "TLSSetup": {  
        "method": "dominoMicroCA",  
        "CADisplayName": "Demo CA",  
        "CAOrgName": "NotesLab",  
        "CAKeyType": "ECDSA",  
        "CAExpirationDays": 1096,  
        "orgName": "NotesLab",  
        "TLSKeyType": "RSA2048",  
        "certExpirationDays": 120  
    }  
}
```

# Manual Certificate Operations

- 1. CertMgr processes submitted requests and creates
  - Private key ( RSA or ECDSA)
    - Saved locally encrypted for assigned servers
- CSR (**Certificate Signing Request**) signed by private key → PEM
- 2. Admin copies CSR to CA
- 3. Admin imports certificate & chain ( PEM ) back
- Paste full chain in any order and submits the form again
- Duplicate certs are ignored
- Missing intermediate certs and root are automatically added from “Trusted Roots” in **certstore.nsf**

Submit Request Save & Close Cancel

TLS Credentials

Main | Security/Keys | Manual | Comments

Main

Status:

Host names: www.notes.lab

Servers with access: notes-lab-01/Srv/NotesLab

Status:

Certificate expiration:

Certificate renew date: Manual

Certificate provider: Manual

Key type: ECDSA

Curve name: NIST P-384

Automatically renew: 30 days before expiration

Keyring file:

Submit Request Copy CSR Paste Certificate Edit

Submit Request Save & Close Cancel

TLS Credentials

Main | Security/Keys | Manual | Comments

Main

Status: Waiting

Host names: www.notes.lab  
notes-lab-01/Srv/NotesLab

Main | Security/Keys | Manual | Comments

Paste - Certificates & Roots (PEM)

Copy - CSR (Certificate Signing Request)

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBijCCARACQAwVjELMAkGAIUEBhMCVVMxCzAJBgNVBAgMAk1B
DAZBb3NOb24xETAPBgNVBAoMCE5vdGVzTGFiMRYwFAYDVQQDDA13
bGFiMHYwEAYHKcZIzj0CAQYFK4EEACIDYgAEpSQ0gM/28q22Yycq
```

# Nash!Com Lab CA for testing

- Manual CA operations require a CA for testing
- Nash!Com CA runs on a Linux server listening on a TCP/IP port
  - Based on OpenSSL C code
- Mail-in database with pre-delivery agent sends HTTPS request to CA
- Send your **CSR** to [lab-ca@nashcom.de](mailto:lab-ca@nashcom.de) with subject: **cert**
- You receive a **certificate** and **chain** in reply!
  - Root certificate is not included in reply – like for most other CAs
    - Import your cert first and see the missing root
    - In the second step let's import the root

# Automated Certificate Management

[Home](#) / [DOMINO-I-12](#) / [New idea](#)

- Support for Let's Encrypt
  - ACME protocol V2 (**RFC 8555**)
  - Automatic Certificate Management Environment
- Free of charge SSL/TLS certificates
- Fully integrated into **certstore.nsf & CertMgr**
- Easy to deploy
- Automatic certificate update (request) and deployment (reload on server)

179  
VOTE

Include Support for Let's Encrypt

see <https://midpoints.de/de-solutions-LE4D>



Guest • Jul 14 2018 • Planning to implement



# ACME HTTP-01 Challenges

- How it works
  1. ACME server sends a challenge to ACME client
  2. ACME server will ask via in-bound HTTP port 80 for the “secret” at a well-known URL
- DSAPI Filter “certmgrdsapi” needs to be enabled in server doc / internet site !!
  - Tip: **load certmgr -c** adds the DSAPI filter to server doc – Internet sites need to specify manually
- If server is configured to only allow authenticated connection configure public URL
  - Notes.ini: HTTPPPUBLICURLS=/.well-known/acme-challenge/\*:/redir.nsf/\*:/MFASetup\*
- Again: **Inbound HTTP port 80 required!**
- If the server is not reachable by the ACME server (e.g. Let's Encrypt), the challenge fails !!!
  - Tip: Inbound connection can be a proxy connection

# ACME Provider Let's Encrypt – included in template

- Let's Encrypt Staging
  - <https://letsencrypt.org/docs/staging-environment>
  - Should always be used for first steps testing connectivity
  - Provides the same functionality like Let's Encrypt production
  - Much higher limits for certificates and errors
- Let's Encrypt production
  - <https://letsencrypt.org>



# Additional tested ACME Providers

- ACME is a standard supported by more providers
- New ACME provider can be added using their published directory URL

- ZeroSSL

- <https://zerossl.com>
  - Requires registration + external account binding (EAB)



- BuyPass

- <https://buypass.com>



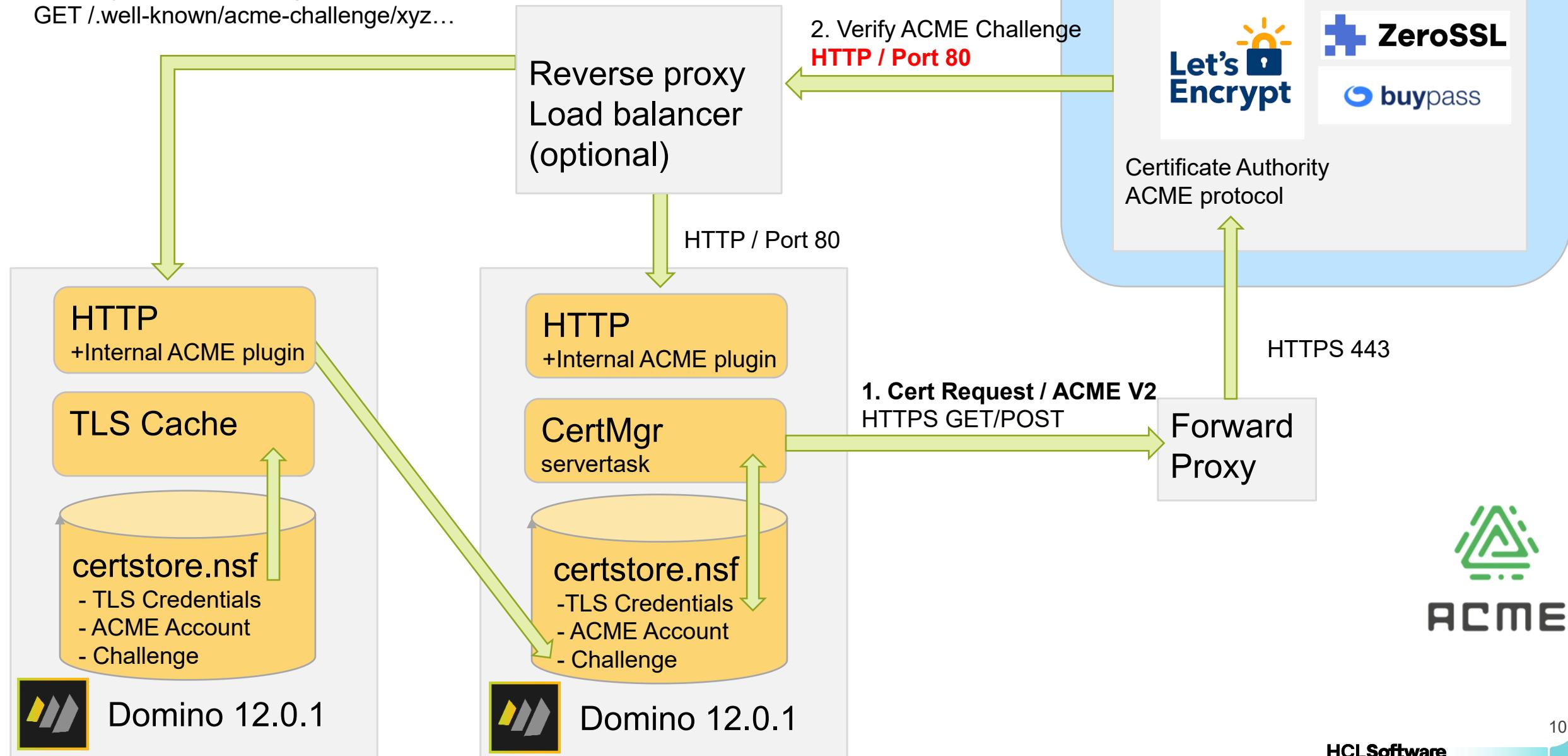
- SSL.Com (requires V12.0.1)

- <https://www.ssl.com>
  - Requires registration + external account binding (EAB)



## 2. Verify ACME Challenge HTTP / Port 80

GET /.well-known/acme-challenge/xyz...



# ACME HTTP-01 Challenges

- How it works
  1. ACME server sends a challenge to ACME client
  2. ACME server will ask via in-bound HTTP port 80 for the “secret” at a well-known URL
- DSAPI Filter “certmgrdsapi” needs to be enabled in server doc / internet site !!
  - Tip: **load certmgr -c** adds the DSAPI filter to server doc – Internet sites need to specify manually
- If server is configured to only allow authenticated connection configure public URL
  - Notes.ini: HTTPPPUBLICURLS=/.well-known/acme-challenge/\*:/redir.nsf/\*:/MFASetup\*
- Again: **Inbound HTTP port 80 required!**
- If the server is not reachable by the ACME server (e.g. Let's Encrypt), the challenge fails !!!
  - Tip: Inbound connection can be a proxy connection

# ACME HTTP-01 Challenge & Troubleshooting

- Port 80 inbound needs to be open unauthenticated for ACME Challenge
  - GET `/.well-known/acme-challenge/xyz...`
  - First request needs to be port 80!
  - ACME challenge request can be redirected
  - All HTTP servers can respond to all ACME challenges
- **No tickets for ACME HTTP-01 challenge issues should be escalated without checking all steps in troubleshooting guide!**
  - [https://github.com/HCL-TECH-SOFTWARE/domino-cert-manager/blob/main/docs/troubleshooting\\_acme\\_challenges.md](https://github.com/HCL-TECH-SOFTWARE/domino-cert-manager/blob/main/docs/troubleshooting_acme_challenges.md)
  - Very detailed guide with links for additional troubleshooting self services
  - Living document in HCL GitHub repository