#dominoforever

Hands On Domino on Docker & K8s

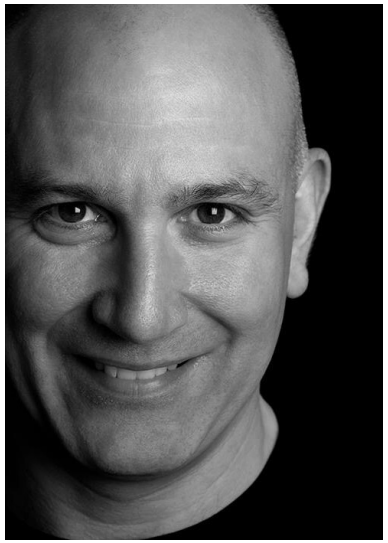
Daniel Nashed, Nash!Com

Special Guest: Thomas Hampel, HCL

Nash!Com
Communication Systems

About Daniel Nashed

- Nash!Com – HCL Business Partner
- Focus: Cross-Platform C-API, HCL Domino® Infrastructure, Administration, Integration, Performance, Security, Troubleshooting and HCL Traveler

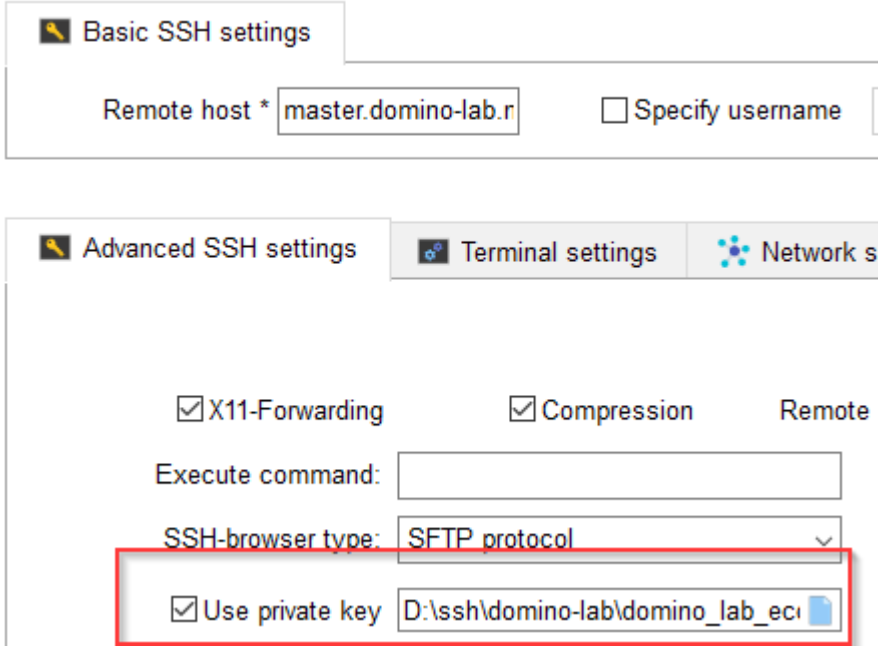


HCL Lifetime Ambassador

- Author: Domino on Linux®/UNIX® Start Script
- Co-Author: Domino Docker Community GitHub Project

Before we start …

Linux Security



- No our Lab isn't a Linux best practices installation
- The machine has no root password but SSH key ..
- Still there are more paranoid ways to do it …
  - You will see a lot of failed login attempts by password
- See blog for details
  - https://blog.nashcom.de/nashcomblog.nsf/dx/paranoid-ssh-configuration-3fa.htm
- DNUG uses password and TOTP token for Linux server
  - This even allows us to login users once if we give them a time code
    - Already tested with HCL Sametime support ;-)

Nash!Com
Communication Systems

What's going on with Docker?

- Docker was innovator but now other projects took the lead
- Kubernetes (K8s) stopped supporting Docker a container run-time
- Even Docker CE 20.10 and Docker Desktop 3.0 has been just releases, the more interesting project is “Podman”
- Podman has interesting new integration “Podman Play” allowing to bridge configurations for K8s YML configurations
- Podman can be used as a almost 1:1 replacement for Docker functionality
  - We are fully supporting Docker and Podman in our GitHub Project
- Whatever software we use it is always about **IMAGES** and **CONTAINERS**

Lab Instructions & Password

- You find the lab instructions here
- https://github.com/nashcom/domino-lab
  - All important commands are prepared
  - Please use the Docker instructions only!
- Password for accessing
  - **registry.domino-lab.net**
  - **domino4ever8**

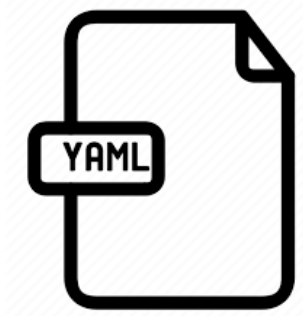
Git

- A tool you should know today!
- Many software developers and companies use Git to manage their source code
- But also GitHub is a platform where you find most of the software projects
  - We can't avoid them even they are owned by Microsoft
- 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

{JSON} – A standard you can't avoid

- https://www.json.org
- Most configuration files today are in JSON
- It replaces XML in many cases
- The most popular parser on Linux:
  - JQ – https://stedolan.github.io/jq/
    - Very powerful and included in the Linux distributions

YAML – Another standard you can't avoid



- 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 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/
    - **yq** is what **jq** is for JSON
    - Also allows to convert 1:1 between JSON and YAML!

XML –> JSON → YAML

XML	JSON	YAML

<Servers>
<Server>
<name>Server1</name>
<owner>John</owner>
<created>123456</created>
<status>active</status>
</Server>
</Servers>
	{
	Servers: [
	{
	name: Server1,
	owner: John,
	created: 123456,
	status: active
	}
	]
	}
		Servers:
		- name: Server1
		owner: John
		created: 123456
		status: active

- Source: https://developer.ibm.com/technologies/containers/tutorials/yaml-basics-and-usage-in-kubernetes/
- Additonals info → Great video with all you need to start: https://www.youtube.com/watch?v=1uFVr15xDGg

Docker Installation

yum & Required Packages

- yum is used to install software packages
  - It connects to our software repository
- **yum install -y net-tools yum-utils**
  - -y means yes to prompts
  - net-tools, yum-utils, and net-tools → important tools
- **yum update**
  - Updates existing packages and kernel to the current version

Install Docker Community **E**dition **20.10**

- New version works better with CentOS 8 – No more firewall issues
- One shell script installs Docker
  - **curl -fsSL https://get.docker.com -o get-docker.sh | bash**
  - Note: Podman install would be just "**yum install -y podman**"
- Enable (auto start) and start the Docker Service
  - **systemctl enable --now docker**
- Allow this host to forward/route IP traffic and restart the network
  - **echo net.ipv4.ip_forward=1 >> /etc/sysctl.conf**
  - **systemctl restart network**
  - Required because Docker adds it's own Docker network interfaces → else no outside communication

Test your Docker Installation

- Run the Docker Hello World image to verify the Docker installations is OK
  - **docker run hello-world**

- Run CentOS latest interactive (container is removed afterwards)
  - **docker run --rm -it centos:latest bash**

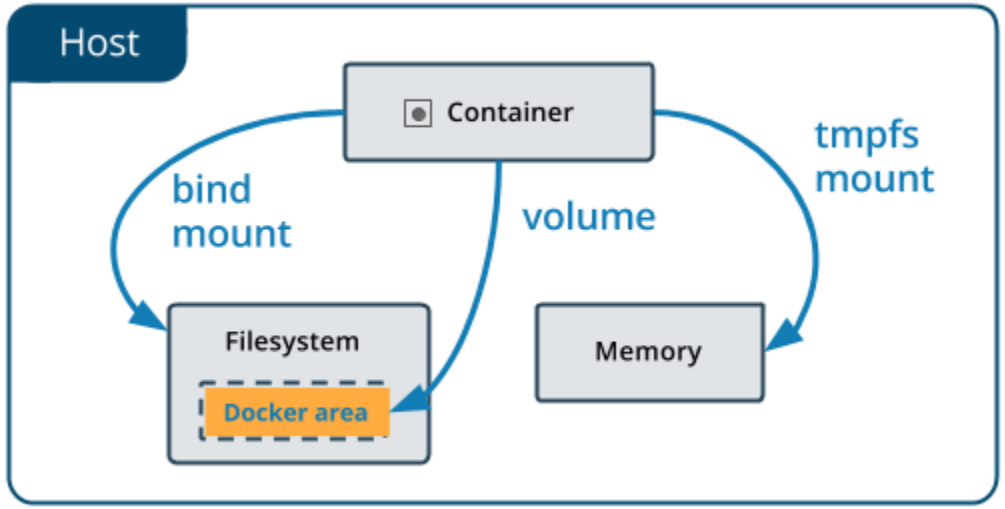
- In both cases Docker pulls down the image from Docker registry/hub and creates a new container and runs it

Working with Docker

- Important commands
  - **docker images** → lists all locally available images
  - **docker ps** → shows all running containers
  - **docker ps -a** → also shows stopped containers
  - **docker run** → creates container from image and starts it
  - **docker start/stop** → starts/stops existing containers
  - **docker inspect** → shows detailed information for a container
  - **docker volume ls** → lists existing volumes
  - **docker volume rm** → removes a volume
  - **docker exec** → executes a command inside a container

Docker Volumes

- By default all data is stored in the container
  - For applications with local storage requirements, this does not work well
  - Therefore Docker supports "**volumes**" which are mapped in to 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

- Multiple options
  - 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 local volume
- **docker volume rm my-vol**
  - Removes volume!
- **docker system df**
  - Checks for used/free space
- **docker system prune**
  - WARNING! This will remove:

    - all stopped containers

    - all networks not used by at least one container

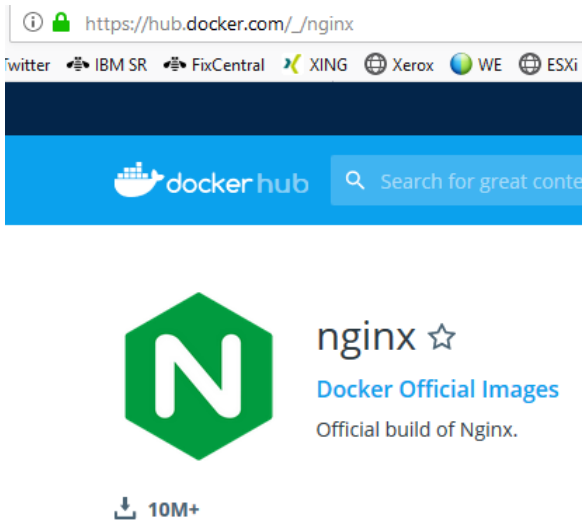
    - all dangling images

    - all dangling build cache

Example: NGINX Reverse Proxy, Web Server, ..

- Available as Docker image from Docker Hub
  - https://hub.docker.com/_/nginx

- About
  - Nginx (pronounced "engine-x") is an open source reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols,
    as well as a load balancer, HTTP cache, and a web server (origin server).
    The nginx project started with a strong focus on high concurrency, high performance and low memory usage. It is licensed under the 2-clause BSD-like license.

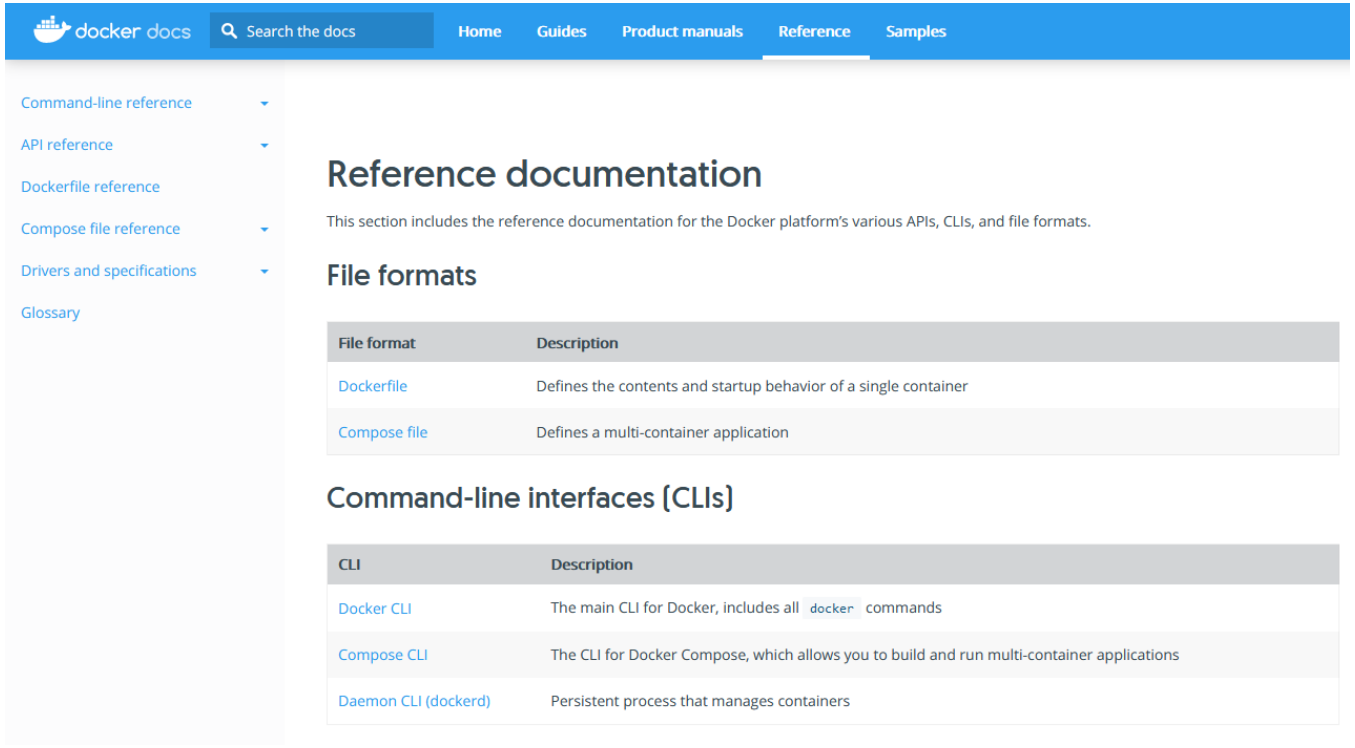


NGINX Configuration

- Hosting local Data
  - **docker run --name hclsoftware -p 7777:80 -v /local/software:/usr/share/nginx/html:ro -d nginx**
    - Shares the local /local/software directory on Docker host's IP port 7777
  - **docker run –name nginx -v /host/path/nginx.conf:/etc/nginx/nginx.conf:ro -d nginx**
    - Mounts a custom configuration into the container via volume command

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.loc



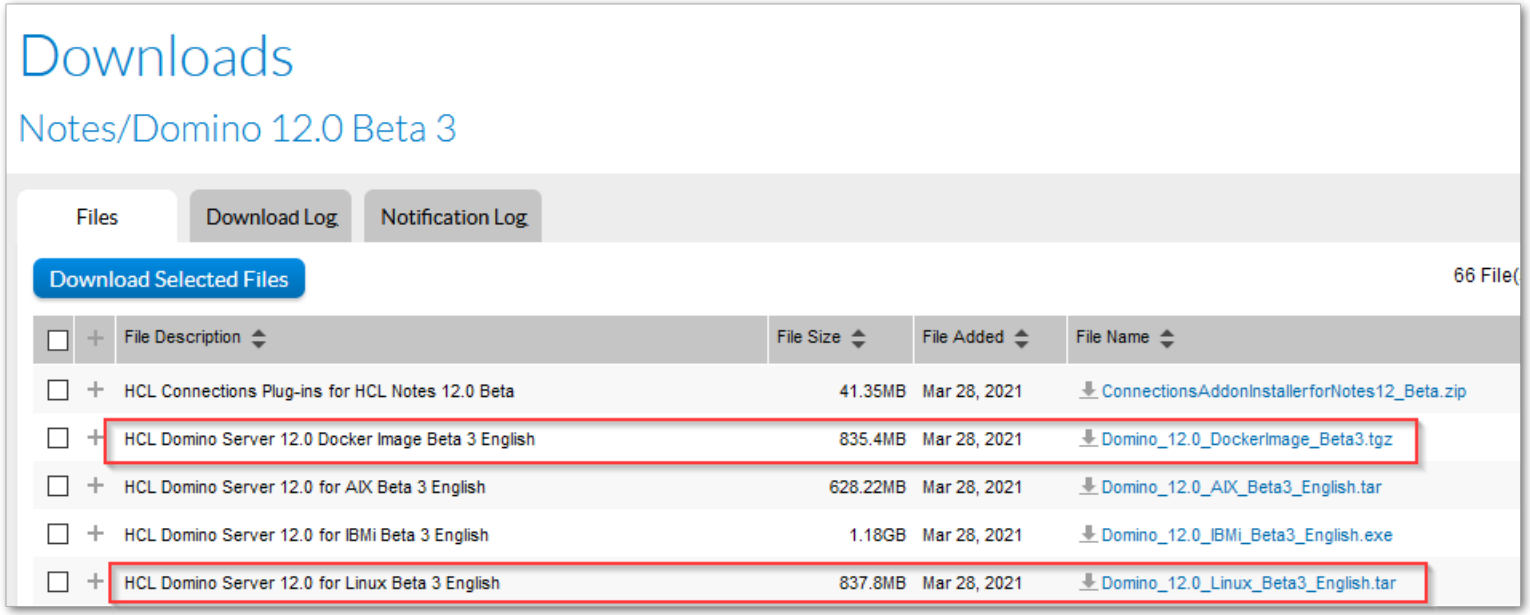
Domino on Docker

Install Editor of your choice

- Not everyone is a **vi** fan ..
  - You should at least know now to edit simple lines, safe and escape ...
- Recommended editors – Already installed in lab environment
  - Nano
    - **yum install -y nano**
  - MC Edit from Midnight commander
    - **yum install -y mc**
- Export editor in your shell to define the editor for our scripts
  - **export EDIT_COMMAND=nano**

Official HCL Image

- Available for Domino 11.0.1 and higher
  - https://help.hcltechsw.com/domino/11.0.1/admin/inst_dock_domino_overview.html
- Needs to be uploaded to the Docker host
  - **Example: docker load --input Domino_12.0_DockerImage_Beta3.tgz**
- HCL Domino V12 Beta 3 available on Flexnet
  - Already downloaded for you



Downloads

Notes/Domino 12.0 Beta 3

Files | Download Log | Notification Log

Download Selected Files

66 File(

File Description	File Size	File Added	File Name
HCL Connections Plug-ins for HCL Notes 12.0 Beta	41.35MB	Mar 28, 2021	ConnectionsAddonInstallerforNotes12_Beta.zip
HCL Domino Server 12.0 Docker Image Beta 3 English	835.4MB	Mar 28, 2021	Domino_12.0_DockerImage_Beta3.tgz
HCL Domino Server 12.0 for AIX Beta 3 English	628.22MB	Mar 28, 2021	Domino_12.0_AIX_Beta3_English.tar
HCL Domino Server 12.0 for IBMi Beta 3 English	1.18GB	Mar 28, 2021	Domino_12.0_IBMi_Beta3_English.exe
HCL Domino Server 12.0 for Linux Beta 3 English	837.8MB	Mar 28, 2021	Domino_12.0_Linux_Beta3_English.tar

Import Docker Image

- Import image to your Docker host via “docker” command
  - **docker load --input Domino_12.0_DockerImage_Beta3.tgz**

    16200ece0bc9: Loading layer [=====>] 156.1MB/156.1MB

    aa88df1bfd40: Loading layer [=====>] 996MB/996MB

    Loaded image: domino-docker:V1200_03252021prod

- Verify the Docker image is on your Docker server

```
docker images

REPOSITORY              TAG                  IMAGE ID          CREATED           SIZE

domino-docker           V1200_03252021prod   4c41d6a16858      4 weeks ago       1.54GB
```

Start HCL Domino Container in Setup Mode

- Command-Line to create a container from an image for setup

```
docker run -it -p 8585:8585 \
--hostname=marvel.csi-domino.com --name domino12 \
--cap-add=SYS_PTRACE  --rm \
-v notesdata:/local/notesdata \
domino-docker:V1200_03252021prod --setup
```

- Details
  - **--cap-add-SYSPTRACE** → important for NSD / gdb
  - **-p** → publishes external ports (sets Linux firewall rules automatically!)
  - **-v** → Maps volume to the container
  - **--rm** → Deletes container on shutdown

Run your first Domino Server

- Command-Line to create a run-time container

```
docker run -it -p 80:80 -p 443:443 -p 1352:1352 \
--hostname=marvel.csi-domino.com --name domino12 \
--cap-add=SYS_PTRACE \
--stop-timeout=90 \
-v notesdata:/local/notesdata \
domino-docker:V1200_03252021prod
```

- Tips
  - In some environments with multiple IP addresses, you might need to switch to host network mode
    - **--network=host**
  - You can use "**docker cp**" commands to copy data from/to the container

Configure a Domino 11 Server with HCL Image

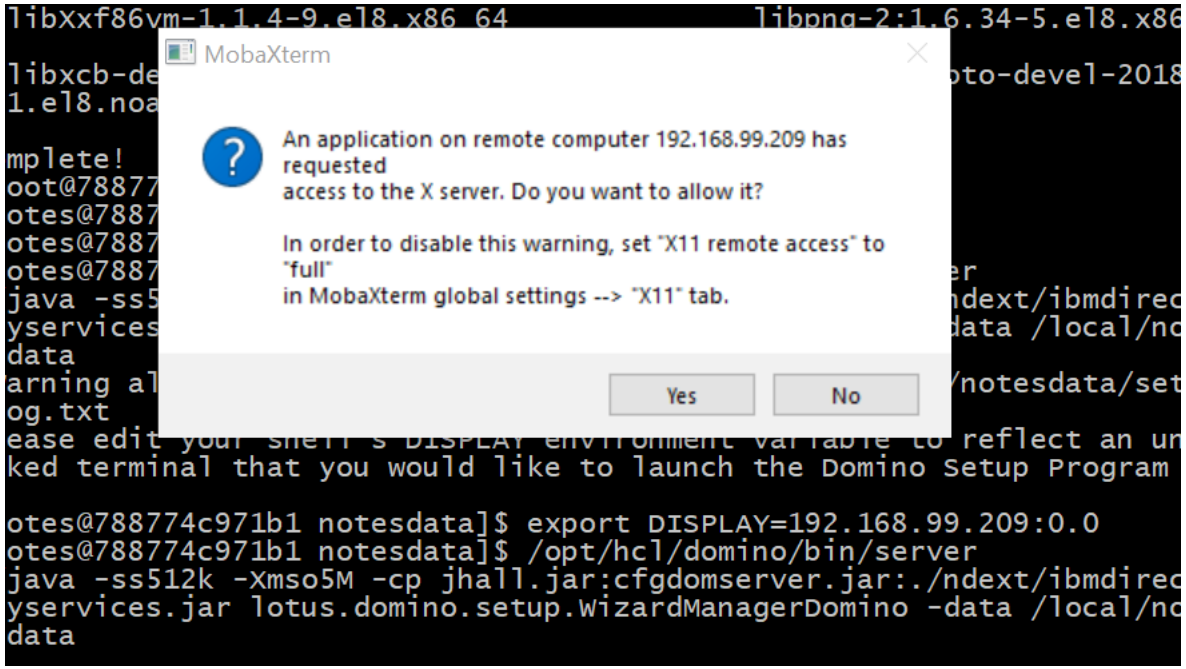
- Server will start in “listening mode”
- Needs <u>remote</u> setup
  - No X window setup, because no X window libs are installed
  - You need to install the “**Remote setup tool**” option on you admin client
    - Tip: Files can be also downloaded from server – see documentation below
  - It's basically the same (Java based) setup launched locally on Windows
  - Documentation “Remote setup”
    - https://help.hcltechsw.com/domino/10.0.1/inst_usingthedominoserversetupprogramremotely_t.html

REF - Plan B: Install X11 & use a remote X11 Server

- Domino setup can leverage X11 on a remote X11 server
  - Just the required X11 libs are missing
  - UBI 8 does only contain limited X11 support without Redhat subscription
  - But the contained libs are sufficient for Domino remote configuration
- Open a root bash into the running container and install missing X11 libs
  - **docker exec -it -u 0 domino12 bash**
  - **yum install -y xorg-x11-apps libX***
  - **exit**

REF - Install X11 & use a remote X11 Server

- Open a normal bash with “notes” user, export your DISPLAY
  - **docker exec -it -u 0 domino12 bash**
  - **export DISPLAY=192.168.96.112:0.0**
- Switch to data directory and run server process to start configuration via X11
  - **cd /local/notesdata**
  - **/opt/hcl/domino/bin/server**
- Tip: MobaXTerm on Windows is very easy to setup
  - No authorization to set. Just prompts when a remote process tries to open a session



Nash!Com
Communication Systems

Interact with your Domino Server

- Normal administration should be performed via Admin client
- You can attach to the Docker container to see the console and interact
  - **docker attach domino12**
  - Exit via CTRL-p + CTRL-q
- Or use a bash into the running container we used earlier for direct Linux accees
  - **docker exec -it domino12 /bin/bash**
    - Return via “exit”
    - Use “-u 0” option for root access
  - Remember: Software you install in a container and files you edit in the container file-system are gone, when you run a new container!

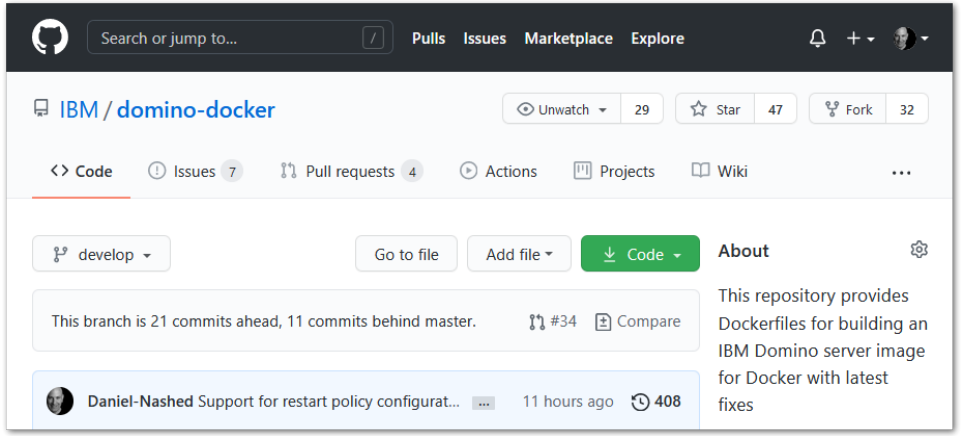
Stop a Domino Server cleanly!

- 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
- Solution
  - Specify a longer shutdown grace period and ensure that the main process will catch the shutdown request to start the Domino shutdown
  - **docker stop –time=120 domino12**
    - Or set –**stop-timeout=90** on the run statement

Docker Compose

- Uses a YAML file to define one or multiple containers
  - They can be joining the same network
  - Sametime meetings is using docker-compose to bring up multiple containers as one service
    - Helm is used for K8s
- 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
- See examples in the Git repository
- We will look into something quite similar in the K8s space after lunch..
  - Podman has “podman play” bridging to the K8s world

Domino Docker Community Image



Nash!Com
Communication Systems

Get the Domino Docker Community Scripts

- Install git software
  - **yum install -y git**
- Create a new directory for your git projects and switch to it
  - **mkdir -p /local/github**
  - **cd /local/github**
- Clone ("download") the official repository locally
  - **git clone https://github.com/IBM/domino-docker**

Download or map Domino Software

- The Open Source Docker Script requires software either locally or on a download location

- Two different modes
  - a.) Store the files locally and have a local NGINX Server provide the software locally
  - b.) Specify central / remote download location

  - In both cases the install process downloads files into the running install image
    - After installation files are removed to keep the image “smaller”

Software <u>local</u> on Docker Host Machine

- Download required files into the software directory
  - e.g. /local/github/**domino-docker/software**

- By default the build process starts a local NGINX server to host files from local software directory
  - Based on a Docker container which is removed afterwards (See part one)

Software on <u>remote</u> File-Server Location

- The build process can also access remote locations
  - We will use a central server
- Edit **DOWNLOAD_FROM** in build.cfg
  - **./build.sh cpcfg**
  - **./build.sh cfg**

    → Specify **DOWNLOAD_FROM=http://registry.domino-lab.net:7777**

Start the “build” Process

- Now that you have configured the script you can start the build process
- A central script “**build.sh**” is used to invoke the build
- Currently implemented images
  - **./build.sh domino**
  - **./build.sh traveler** (builds an image based on the “domino” image)
  - **./build.sh volt** (builds an image based on the “domino” image)
- By default the latest version is used
  - You could choose a specific version by explicitly specifying the version
    - **./build.sh domino 11.0.1 FP3**
    - **./build.sh domino 12.0.0BETA3**

Run your first Domino Server

- Command-Line to create a container from an image
  - **docker run -it -p 80:80 -p 1352:1352 **

    **--hostname=myhost --name mycontainer **

    **--cap-add=SYS_PTRACE **

    **--env-file env_domino **

    -v notesdata:/local/notesdata domino-docker:V1200_03252021prod

- Details
  - **--cap-add-SYSPTRACE** → important for NSD / gdb
  - **--env-file** → passes environment variables to the container
  - **-p** → publishes external ports (internally sets firewall rules automatically!)
  - **-v** → Maps volumes to the container

Domino V12 “One Touch Setup”

One-touch Domino setup

Use one-touch Domino setup to simplify setting up a server.

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.

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.

Using one-touch Domino setup you can:

- 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.

One-touch Domino setup is supported on Domino on Docker, Windows, and UNIX platforms.

Domino V12 “One Touch Setup”

- Domino V12 introduces “One Touch Setup”
  - Brings our **1. automated setup** + **2. Server config.json** from the Community Image into core Domino
  - Cross platform, integrated into core – Not only for containers!
- Two modes
  - a.) **Environment variables** – Designed to use with containers
    - Basic functionality
  - b.) **JSON File** → A lot of additional functionality
    - Combines functionality from remote setup and “config.json” from the Docker Community image
    - Plus more like: Create an ID Vault

Domino V12 “One Touch Setup”

- For basic configuration just use environment variables
- More advanced functionality is available passing a JSON file
  - We will have a detailed example in the K8s workshop part …
- Best starting point is the new V12 documentation
  - https://help.hcltechsw.com/domino/12.0.0/admin/inst_onetouch.html
  - The environment variables are straightforward
    - There is almost a 1:1 replacement with the old setup variables

Kubernetes / K8s

Containers, that'll fix it.

Kubernetes for beginners.

What could go wrong?

O RLY?

DevOops

Nash!Com
Communication Systems

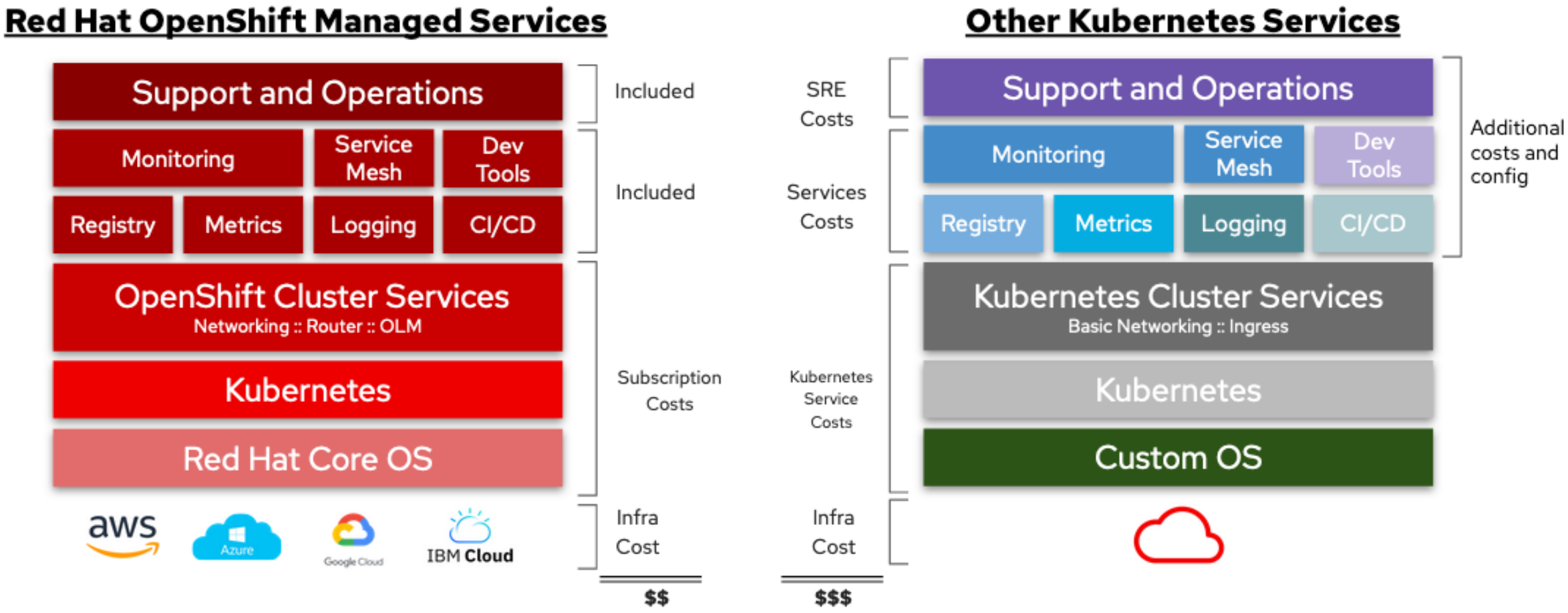
Kubernetes – K8s

- https://kubernetes.io/
- Vanilla K8s is the base for many other distributions like OpenShift, Rancher and others
- There are many projects derived from it and adding functionality

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

OpenShift Stack compared to native K8s



Nash!Com
Communication Systems

OpenShift Installation

Tip: Post from an old IBM friend about
http://heidloff.net/article/running-managed-openshift-on-premises/

- This takes a while …
- And needs a lot of resources in ramp up
  - Mimium install: 3 Hosts x 4 CPU cores, 16 GB RAM, 100 GB Disk

Installation progress: openshift

Started on
11/21/2020, 6:31:18 PM

Status
Installing

Installation in progress: 67%

Download kubeconfig

Bare Metal Inventory

Hostname	Role	Status	Discovered At	CPU Cores	Memory	Disk	(3)
openshift1 *	Master (bootstrap)	Installing 4/7	11/21/2020, 5:35:11 PM	4	16.00 GB	121.00 GB	
openshift2 *	Master	Installing 5/7	11/21/2020, 5:57:38 PM	4	16.00 GB	120.00 GB	
openshift3 *	Master	Installing 5/7	11/21/2020, 6:11:26 PM	4	16.00 GB	120.00 GB	

Cluster Details

OpenShift version
4.6

Base DNS domain
nashcom.loc

API virtual IP
192.168.96.160

UUID
e6c4bc06-3366-4183-a1e8-b1cfbd8c9271

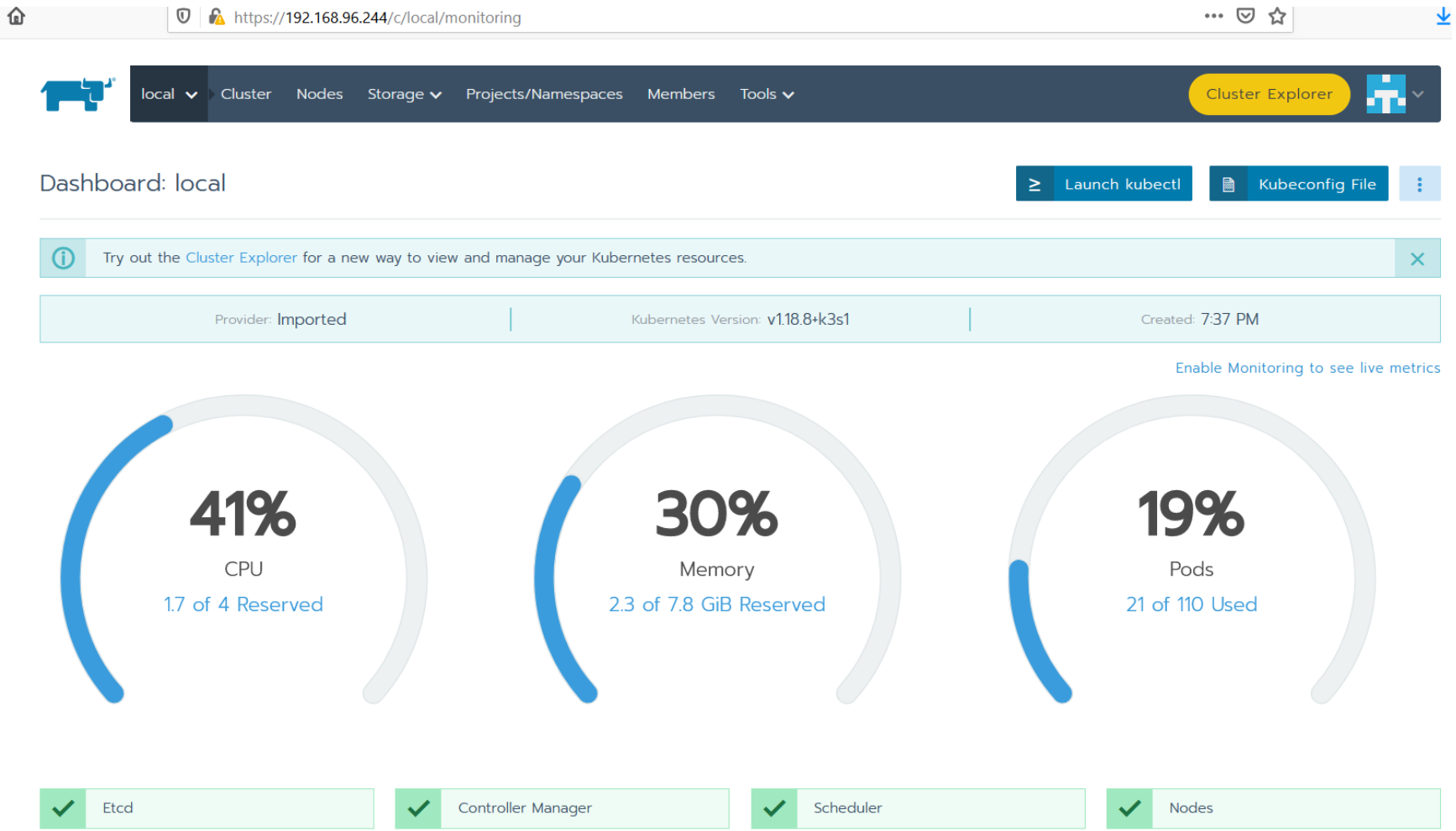
Cluster network CIDR
10.128.0.0/14

Cluster network host prefix
23

Nash!Com
Communication Systems

Rancher basic setup on one server with Docker host

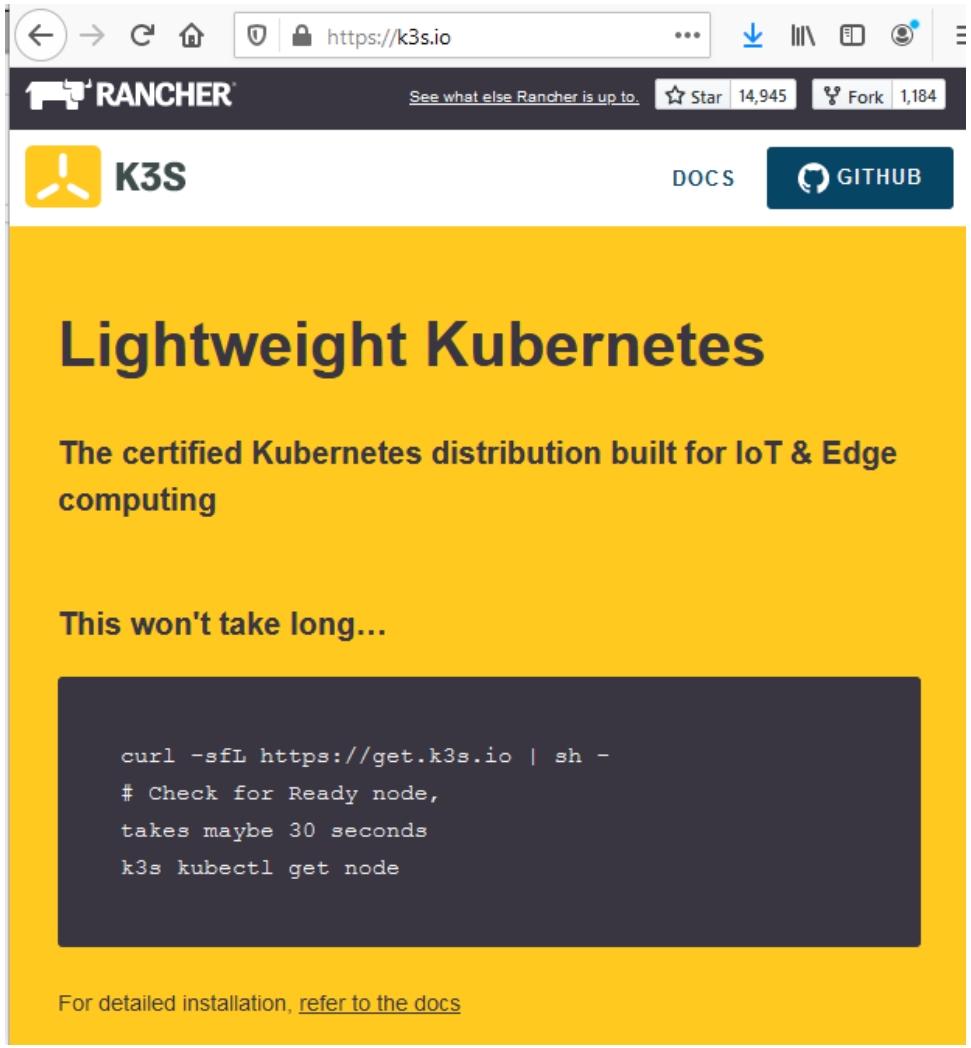
- Very easy to install
- A single server with 8 GB RAM works for small deployment



Nash!Com
Communication Systems

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
- https://k3s.io/
- A single command installs the K3s
  - **curl -sfL https://get.k3s.io | sh -**
- Check installation
  - **k3s kubectl get node**



Access K8s from another machine

- Copy **/etc/rancher/k3s/k3s.yaml** into home dir on another machine
  - Linux/Mac: **~/.kube/config** or **export KUBECONFIG=/etc/rancher/k3s/k3s.yaml**
  - Windows **set KUBECONFIG=c:/k3s.yaml**
- Install **kubectrl** for your platform
  - https://kubernetes.io/docs/tasks/tools/install-kubectl/

Working with kubectrl

- Kubectrl is the management command-line tool for K8s
  - **kubectrl apply -f domino12.yml** → deploys for example a “pod”
  - **kubectr get all** → shows current information
  - **kubectrl exec -it pod/domino12 --bash** → opens a shell inside the container

    kubectrl delete -f domino12.yml → removes what you have deployed
- Similar to docker-compose .yml files are used to describe pods, services, storage, network, etc
  - The syntax is not the same, but for example “podman play” can create those files from running “pods”

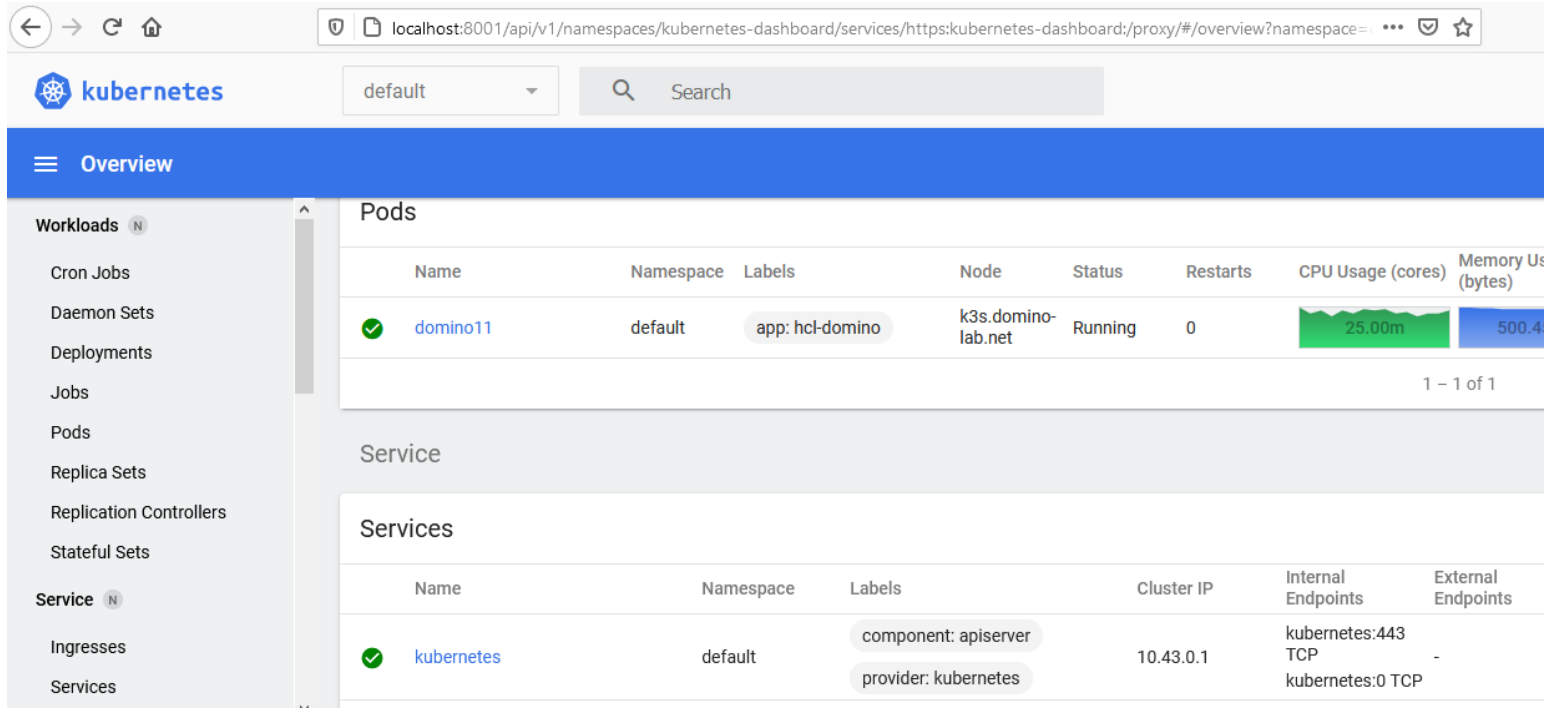
Plain Kubernetes does not have a “Image” registry on it's own

- Platforms like OpenShift, Rancher etc. include registries
- There are many different registries available
  - Some are Docker images on their own
    - For example the Docker **registry2** → easy to install and lightweight
      - https://docs.docker.com/registry/
    - **Sonatype Nexus 3** has a registry included and can serve many other repository types
      - https://help.sonatype.com/repomanager3/formats/docker-registry
      - https://help.sonatype.com/repomanager3
  - GitHub started their own “container registry” which can host public and private images

K8s Dashboard

- Graphical interface
  - Runs inside K8s
- Easy to install
  - But you have to understand how to run it remotely
  - Install the dashbaord
    - https://rancher.com/docs/k3s/latest/en/installation/kube-dashboard/
  - Run kubectr connection proxy on your emote machine
  - Launch the dashbaord locally via proxy connection

    http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/



Nash!Com
Communication Systems

Minikube – Referenced by HCL as a test environment

- https://minikube.sigs.k8s.io/docs/start/
  - Requires 2 CPU cores minimum – doesn't start with less!
  - Requires Docker, Podman installed – detects what you have installed and automatically uses it
- Easy Installation steps
  - curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  - sudo install minikube-linux-amd64 /usr/local/bin/minikube
  - yum install conntrack
  - minikube --driver=none start
- Check installation
  - minikube kubectl get all

Minikube Dashboard

- Run “minikube dashboard”
- Get kubconfig and run proxy locally
  - kubctrl proxy
- Launch dashboard in browser with HTTP
  - http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/prox

New – Domino Container Start Script

- Replaces the management script in our Domino Docker Git project
  - Part of Nash!Com start script, integrated into the Domino Docker project
- Similar options you know from the Domino on Linux start script
  - One stop shopping for all your operations
  - Includes Podman systemd integration
- Custom image build support
- Let's have a look …
  - Hands On after the break with Domino V12 Beta 3

PRODUCTS & PLATFORMS
by HCL Technologies

Domino V12 Beta 3

Nash!Com
Communication Systems

Automatic Certificate Management

Automatic Integrated Certificate Management

- First early code drop to get feedback
- Currently implemented
  - Let's Encrypt® certificate requests leveraging “HTTP” challenges
  - New servertask “certmgr” and database “certstore.nsf”
- Certificates and keys are stored in PEM format
  - Kyr-File format is still required in this early stage
    - And will remain as an option for older servers
- Basic flows are implemented
  - Will evolve over code drops – also based on your detailed feedback!

Technology used

- Native Servertask (C/C++)
- Leverages existing and new Notes security APIs
- Implements Let's Encrypt uses ACME protocol V2 (RFC 8555)
  - **AMCE** = **A**utomatic **C**ertificate **M**anagement **E**nvironment
  - Own HCL implementation leveraging standards like
    - JSON, LibCurl, JWS, OpenSSL, Notes crypto including PEM, RSA and ECDSA keys ...
- Already designed for “automation”
  - If your server is available on HTTP (port 80) and HTTPS (port 443) and has an DNS entry, you can create your first certificate with one command today

Standard Scenario - Domino V12 with inbound HTTP(S)

- Certmgr servertaks and certstore.nsf on the same Domino V12 server
- DNS pointing to your server
  - DNS Domain like *.csi-domino.com pointing to the server
    - Allows to test with any host name ;-)
- Certmgr supports outbound proxy connections including authentication
- Inbound HTTP(S) connection can use an incoming proxy/load-balancer
- Allows automatic setup if DNS and hostname matches

Architecture Diagram

Let's Encrypt / ACME Flow and components

```
                                        ////////////\\\\\\\\\\
                                        |  Intranet/Internet |
                                        \--------------------/
+------------+                                     |
|  ACME CA   | II. Challenge                       | [HTTPS/443]
|  Server    |----[HTTP/80]----+     +----------+
+------------+       GET       |     |
      ^                        v     V
      |                     +------------+
      |                     |ReveseProxy/|
      |                     |LoadBalancer|
      | I. Cert Request     +------------+
      |                           |
      | [HTTPS/443]               | [HTTPS/443] (HTTP/80 for Challenge)
      |  GET/POST                 V
      |  ACME V2            +-------------+                  CertStore
      |                     |   Domino    |              +---------------+
      |                     |HTTP + DSAPI |<---[NRPC]----| A. X.509/KYR  |
      |      +-------+      +-------------+              | B. Challenge  |
      +------| Proxy |<-----|Let's Encrypt|<---[NRPC]--->|               |
             +-------+      |    Task     |              |---------------|
                            | (KyrTool)   |<---[NRPC]--->| C. LE Account |
                            +-------------+              | D. Proxy Auth |
                                                   *)    +---------------+
```

Components

*) Connection between Domino, LE and CertStore could be local or NRPC

Domino HTTP and LE could be on separate server and just need a common CertStore.

(A) X.509 today in kyr file
(B) Challenge needed to verify request
(C) LE used to authenticate with ACME CA
(D) Proxy account needed for outgoing communication

Flow

1. LE creates account (C) with ACME server
2. LE creates private key and writes it to CertStore (A)
3. LE creates CSR and sends it to ACME CA *)
4. LE puts received challenge (B) in CertStore
6. ACME server requests challenge on port 80 to verify
7. Domino HTTP replies with challenge (B) from CertStore
8. LE receives certificate including and writes it to CertStore (A)

HTTP (and INET tasks) read X.509 from CertStore (A)

*) Proxy communication uses Proxy user (D)

Nash!Com
Communication Systems

Questions & Discussion + Special topics?

CentOS Stream

CentOS 8.x Stream

Nash!Com
Communication Systems

CentOS Stream – What's going on?

- There is a lot of confusion
- It's not new information!
- An it isn't as critical as many discuss it
- There is no such thing as “Enterprise level free beer”
  - You always have to pay a price or take a “risk”
- CentOS Stream is a stable and free enterprise Linux
  - It is the version becoming the next RHEL dot release
  - CentOS 8 & CentOS Stream meet the minimum support requirements
  - But are not an officially tested HCL configuration

Matthew Miller

“Stable” is a loaded word with a lot of different meanings. The changes that will land in CentOS Stream are intended for the next RHEL minor release, and so should not be more drastic than one might expect from that kind of update.

Note that CentOS Linux 8 as a traditional rebuild is still a thing.

OCTOBER 2, 2019

Quote from: Matthew Miller, Distinguished Engineer at Red Hat / Fedora Project Leader

fedora → Red Hat Enterprise Linux → CentOS

Bleeding edge

Enterprise Release
Stable / Commercial

Free & based on stable RHEL

fedora → CentOS Stream → Red Hat Enterprise Linux

Bleeding edge

Ahead of REHL becomes the next RHEL <u>dot</u> release

Enterprise Release
Stable / Commercial

Nash!Com
Communication Systems

Domino V12 Supported Linux Platforms

- New support statement
  - **1. Full HCL tested platforms**
    - Tested and fully support
    - Recommended for enterprise environments
  - **2. Platforms meeting defined minimum requirements**
    - Not tested by HCL
    - Expected to work
    - Tested by the community and partners
    - Standard support for Domino and add-ons

- Minimum Requirements for Domino V12
  - **RHEL/CentOS 7.5 Linux-equivalent OS**
    - kernel 3.10.0-693 x86_64 or higher 3.x kernel
    - glibc 2.17-222 x86_64 or higher
    - libstdc++-4.8.5-28 x86_64 or higher
  - **RHEL 8.0 Linux-equivalent OS**
    - kernel 4.18 x86_64 or higher 4.x kernel
    - glibc 2.28 x86_64 or higher
    - libstdc++-8.2.1 x86_64 or higher
  - Domino V12 Installer will check requirements and print warnings

Domino V12 Supported Linux Platforms

- Fully tested and full supported platforms
  - SLES 12.2, 15.0
  - RHEL 7.5 + 8.0
  - CentOS 7.5
- Some Platforms meeting the minimum requirements for “best effort” support
  - CentOS 8, CentOS 8 Stream, Alma Linux
  - SUSE Leap 15.x
  - Ubuntu Server LTS 20.04
  - Astra Linux, (latest common edition & special edition)
  - Debian 10.x should also work

AlmanLinux – A CentOS 8 Clone

- If you really look for an alternate CentOS clone, have a look at AlmaLinux
  - https://almalinux.org/
- Well done distribution implemented by CloudLinux
  - Still this isn't a “fully tested” distribution
- Also provides a Docker base image
  - dockerfile_alma included in Docker Community project
- Already available in contrast to RockyLinux