

Docker, Kubernetes, Helm

into DevOps methodologies

Baxenergy.com

This document is proprietary to BaxEnergy Italia Srl and contains reserved information and trade secrets: it is supplied in confidence and it is intended solely for the use of the individual entity to whom it is addressed. It should not be disclosed, duplicated, translated or otherwise revealed in whole or in part without the prior written consent of BaxEnergy Italia Srl. All rights are reserved.

Copyright© 2019 BaxEnergy Italia S.r.l. All rights reserved. EnergyStudioPro®, AssetStack and their module are registered trademarks of BaxEnergy Italia S.r.l.

GROW YOUR DEVOPS SKILLS

LEARN HOW TO GET STARTED WITH DEVOPS

Whether you are from a system admin or software developer background, or if you are a recent new graduate or have been in the industry for a while, this book will give you an overview of the DevOps industry and provide you with a practical guide about how to get started with a career in the DevOps domain.

GET YOUR FREE COPY NOW



<https://www.level-up.one/devops-pdf-book/>

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

Contacts: feel free contact us via LinkedIn



Mariano Alesci

DevOps Architect presso BaxEnergy



Lorenzo Vetrano

Scrum Master and Agile Coach

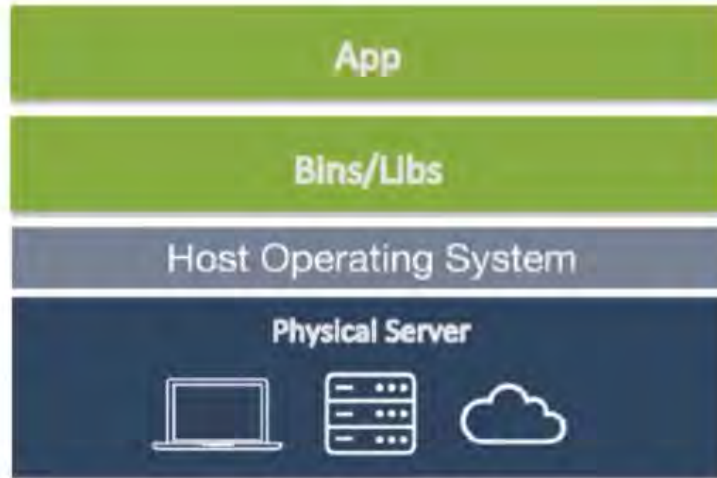
Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

Introduction to Virtualization Technologies

Docker technology

is one implementation of container based
virtualization technologies

Pre-Virtualization World

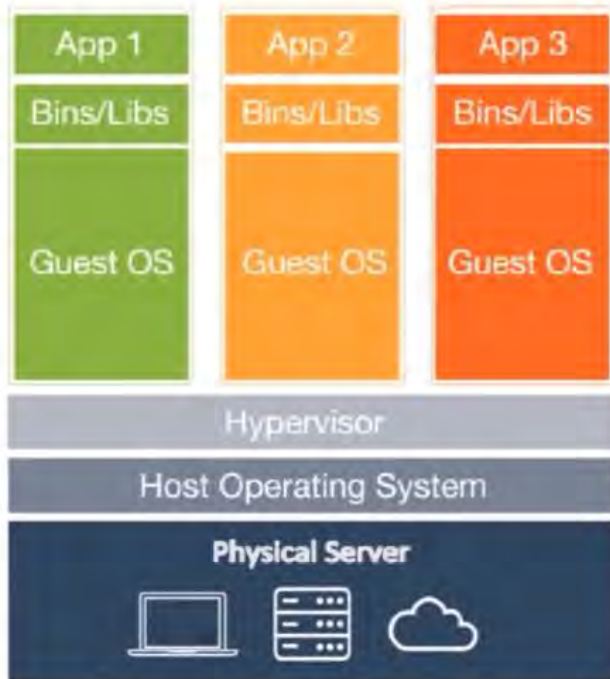


Pre-Virtualization

Problems:

- Huge Cost
- Slow Deployment
- Hard to Migrate

Hypervisor-based Virtualization



Hypervisor-based Virtualization

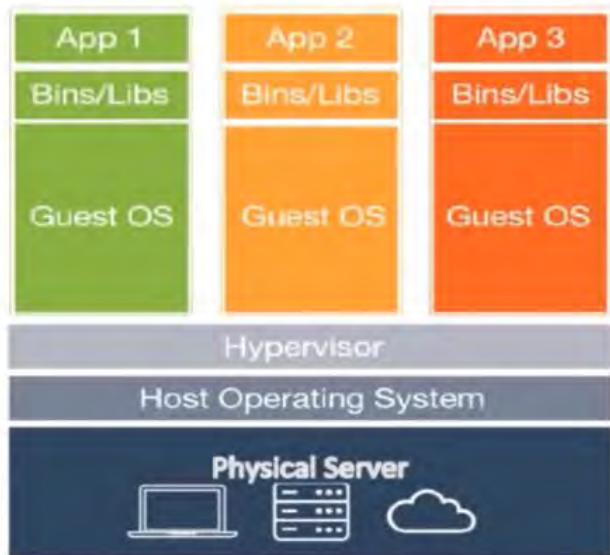
Benefits:

- Cost-Efficient
- Easy to Scale

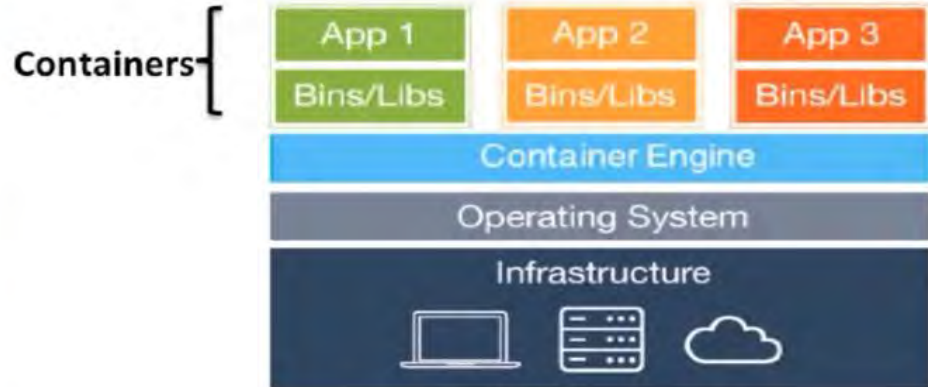
Limitations:

- Kernel Resource Duplication
- Application Portability Issue

Hypervisor-based VS Container-based Virtualization



Hypervisor-based Virtualization

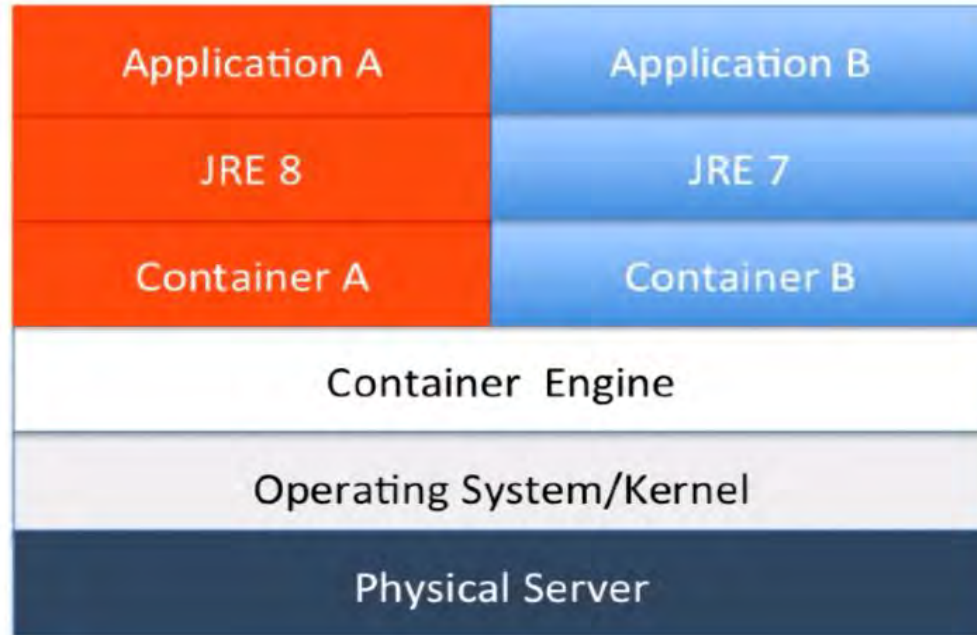


Container-based Virtualization

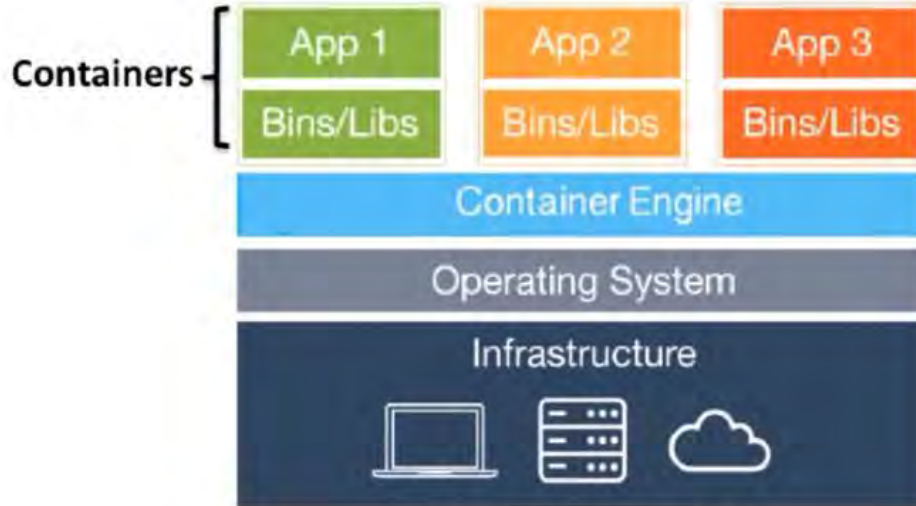
Runtime Isolation

How to run two different Java applications with two different JREs on the same VM?

Runtime Isolation



Container Virtualization



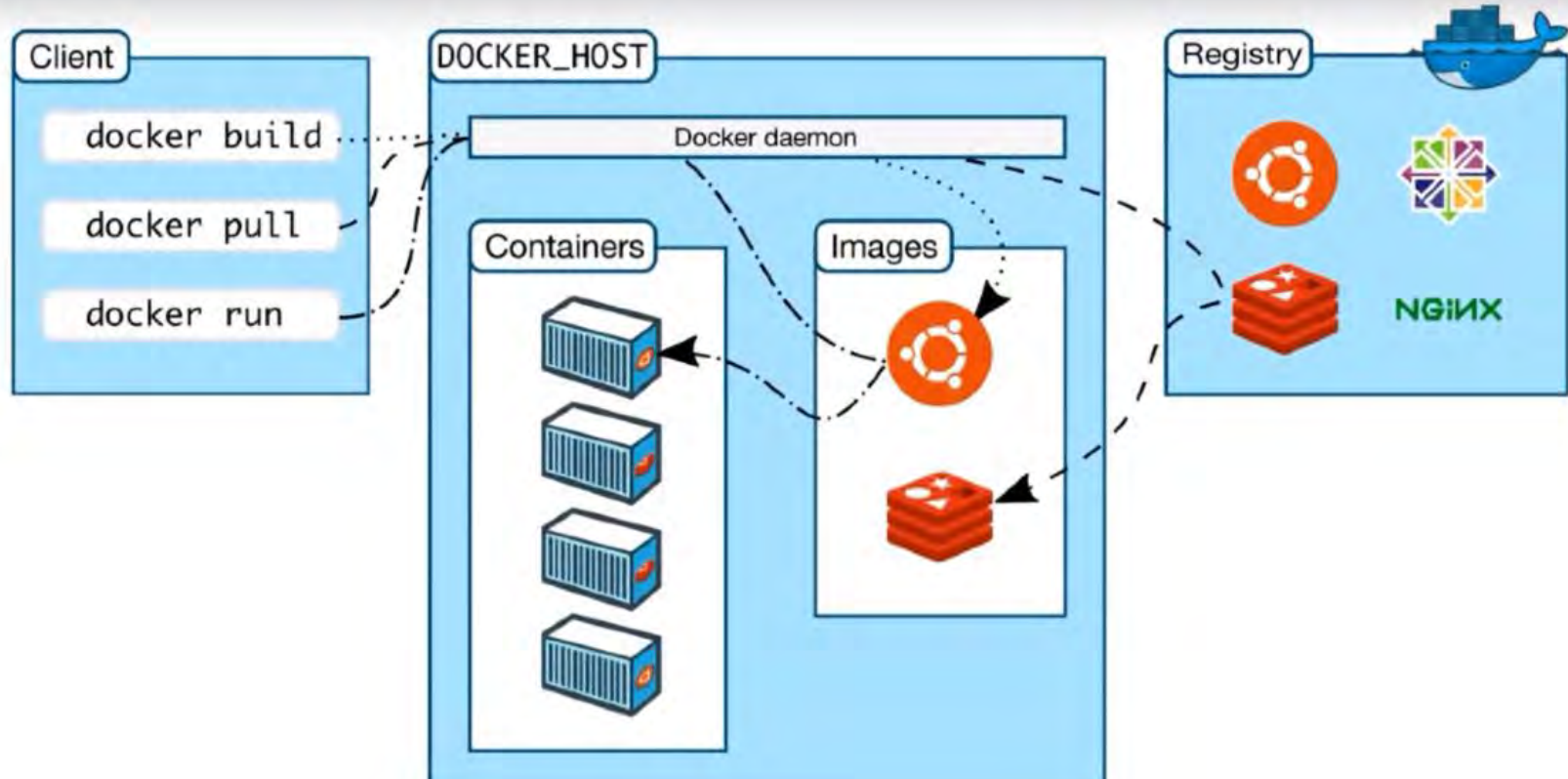
Benefits:

- Cost-Efficient
- Fast Deployment
- Guaranteed Portability

Docker

Client-Server Architecture

Docker: client-server architecture



Docker: client-server architecture

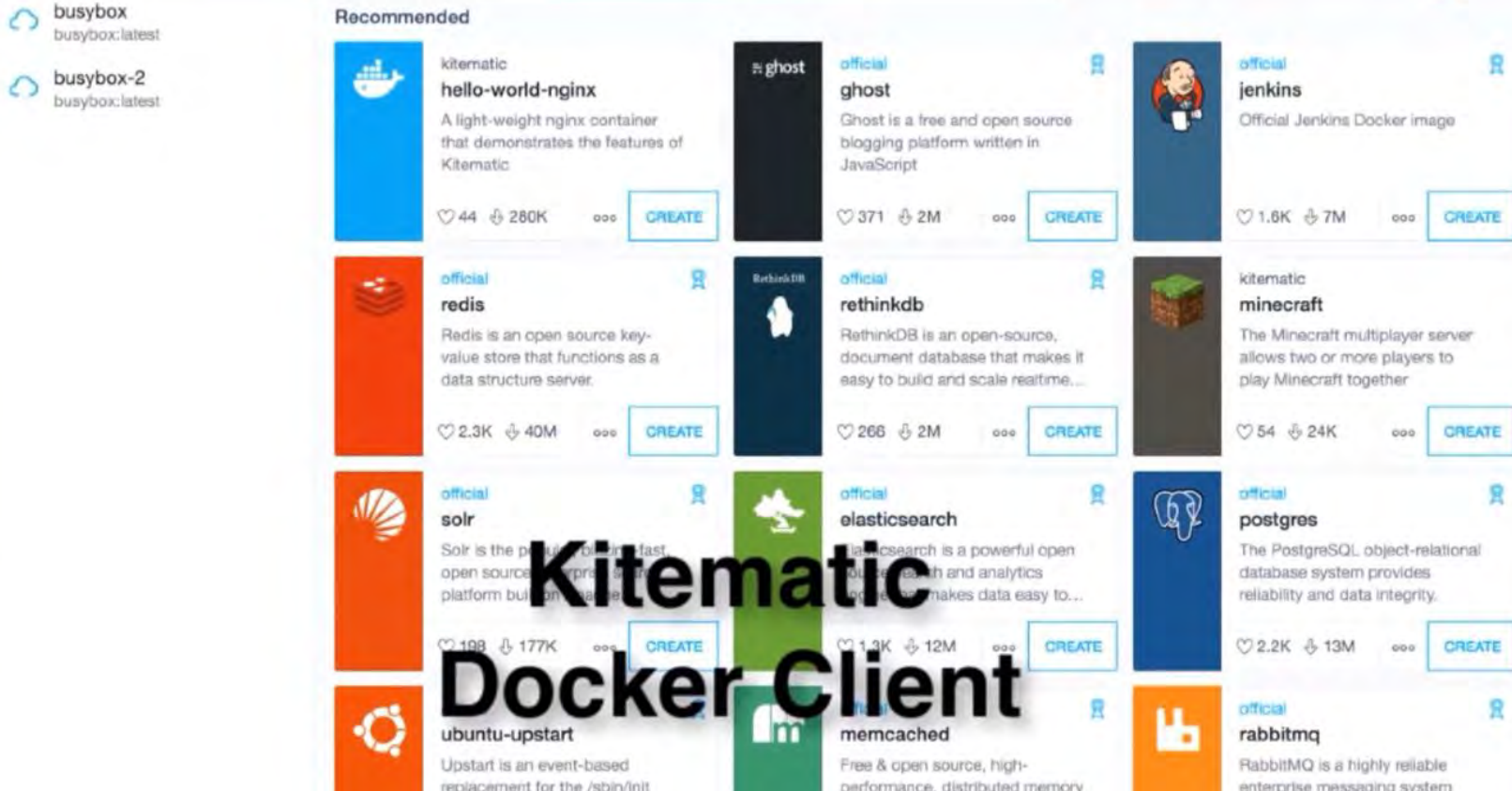
```
bash --login '/Applications/Docker/Docker Quickstart Terminal.app/Contents/Resources/Scripts/start.sh'  
cwei69-mac:~ cwei$ bash --login '/Applications/Docker/Docker Quickstart Terminal.app/Contents/Resources/
```



Command Line Docker Client

```
docker is configured to use the default machine with IP 192.168.99.100  
For help getting started, check out the docs at https://docs.docker.com  
  
cwei69-mac:~ cwei$
```

Docker: client-server architecture



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

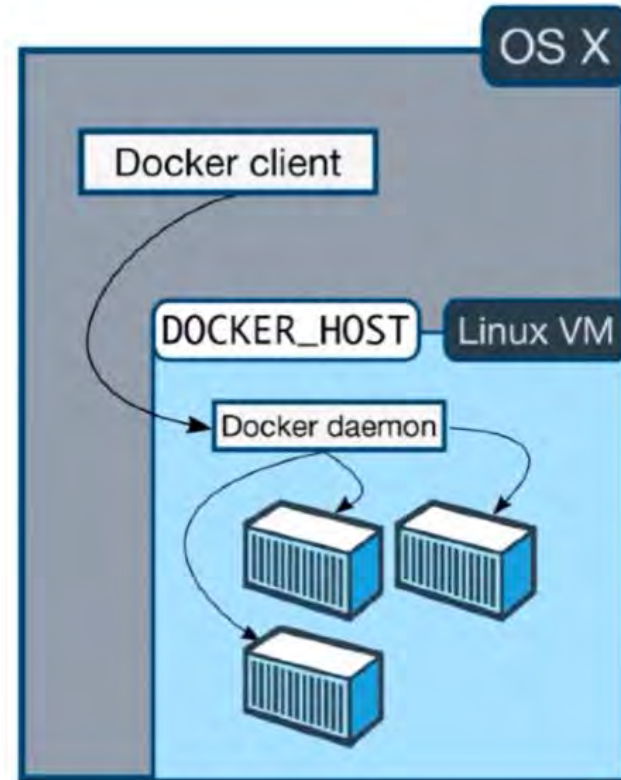
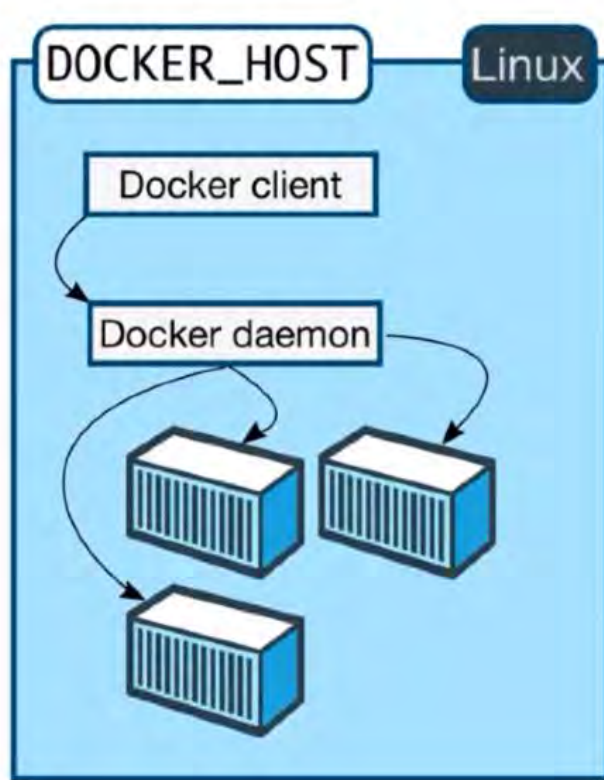


Docker Daemon

Docker Engine

Docker Server

Docker: client-server architecture



Install Docker for Mac/Windows



Install Docker

This lecture applies to you if:

- You are using **Linux**
- Or you are using Mac and your Mac version is **OS X 10.10.3 or newer**
- Or you are using Windows and your Windows version is **Windows 10 or newer**

Otherwise, you can skip this lecture and follow the installation guide of the next lecture.

Docker: installation

The screenshot shows the Docker Engine installation page. The browser address bar displays <https://docs.docker.com/engine/installation/>. The left sidebar contains a navigation menu with the following items: Docker Engine, Docker Overview, Install, Get Started with Docker, Learn by example, User Guide, Admin Guide, Manage a swarm, Secure Engine, Extend Engine, Dockerize an application, Engine reference, and Migrate to Engine 1.10. The main content area is titled "Install Docker Engine" and states: "Docker Engine is supported on Linux, Cloud, Windows, and OS X. Installation instructions are available for the following: On Linux". Below this, a list of Linux distributions is provided: Arch Linux, CentOS, CRUX Linux, Debian, Fedora, FrugalWare, Gentoo, Oracle Linux, Red Hat Enterprise Linux, openSUSE and SUSE Linux Enterprise, and Ubuntu. A note at the bottom of the list says: "If your linux distribution is not listed above, don't give up yet. To try out Docker on a distribution that is not listed above,". On the right side, there is a search bar labeled "Search the docs" and a section titled "On this page" with links to "On Linux", "On Cloud", "On OSX and Windows", "The Docker Archives", and "Where to go after installing".

https://docs.docker.com/engine/installation/

Apps For quick access, place your bookmarks here on the bookmarks bar. [import bookmarks now...](#)

Docker Engine

Docker Overview

Install

Get Started with Docker

Learn by example

User Guide

Admin Guide

Manage a swarm

Secure Engine

Extend Engine

Dockerize an application

Engine reference

Migrate to Engine 1.10

Install Docker Engine

Docker Engine is supported on Linux, Cloud, Windows, and OS X. Installation instructions are available for the following:

On Linux

- [Arch Linux](#)
- [CentOS](#)
- [CRUX Linux](#)
- [Debian](#)
- [Fedora](#)
- [FrugalWare](#)
- [Gentoo](#)
- [Oracle Linux](#)
- [Red Hat Enterprise Linux](#)
- [openSUSE and SUSE Linux Enterprise](#)
- [Ubuntu](#)

If your linux distribution is not listed above, don't give up yet. To try out Docker on a distribution that is not listed above,

Search the docs

On this page

- [On Linux](#)
- [On Cloud](#)
- [On OSX and Windows](#)
- [The Docker Archives](#)
- [Where to go after installing](#)

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond


A screenshot of a macOS Terminal window. The title bar shows 'Terminal' and standard menu items: 'Shell', 'Edit', 'View', 'Window', 'Help'. The status bar at the top right displays system icons (Wi-Fi, battery at 57%, clock at 7:31 PM, and the name 'James Lee'). The terminal window title is 'jameslee -- bash -- 142x37'. The terminal content shows the output of the 'docker info' command, listing various Docker configuration details. The text is as follows:

```
Containers: 0
Running: 0
Paused: 0
Stopped: 0
Images: 0
Server Version: 1.12.0
Storage Driver: aufs
Root Dir: /var/lib/docker/aufs
Backing Filesystem: extfs
Dirs: 0
Dirperm1 Supported: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
Plugins:
Volume: local
Network: bridge null host overlay
Swarm: inactive
Runtimes: runc
Default Runtime: runc
Security Options: seccomp
Kernel Version: 4.4.15-moby
Operating System: Alpine Linux v3.4
OSType: linux
Architecture: x86_64
CPUs: 4
Total Memory: 1.954 GiB
Name: moby
ID: EIJQ:WECM:7FT0:6CTT:LYRU:ISBF:IGPH:AMKH:2M57:DS7J:RSAF:Y7YY
Docker Root Dir: /var/lib/docker
Debug Mode (client): false
Debug Mode (server): true
```

Docker: installation

Most Popular Playgrounds


Playgrounds give you a configured environment to start playing and exploring using an unstructured learning approach. Playgrounds are great for experimenting and trying samples. To learn more about the technology then start with one of our labs.



Visual Studio Code Playground


Full development environment directly in your browser

Explore Playground




Kubernetes Playground

Experiment with Kubernetes in a safe playground



Docker Swarm Playground

Experiment with Docker Swarm in a safe playground




Ubuntu Playground

Experiment with Ubuntu in a safe playground

Explore Playground


<https://katacoda.com>

Newest Playgrounds




Fedora CoreOS Playground

Linux for Containers and Massive Server Deployments




Docker Playground

Use Docker in a sandboxed playground environment



Docker Experimental Playground

Use experimental binaries to try upcoming features



Docker Swarm Mode Playground

Use Docker Swarm Mode and Swarm in a sandboxed playground environment

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

Run our First Hello World Docker Container



Docker: our first container



Learn Create Embed For Vendors For Trainers For Teams For Enterprises Search Claim Your Profile

Container Fundamentals

<https://katacoda.com/lloodse/courses/docker>



Playground

Use Docker in a safe playground environment.

Start Scenario



Install Docker

Installing docker to a VM

Start Scenario



Our first container

Describes how to create your first container

Repeat Scenario



Run a Webapp with Docker

Describes how to create your first container to run a webapp with Docker.

Start Scenario

Important Docker Concepts

Images

- Images are read only templates used to create containers.
- Images are created with the docker build command, either by us or by other docker users.
- Images are composed of layers of other images.
- Images are stored in a Docker registry.

Containers

- If an image is a class, then a container is an instance of a class - a runtime object.
- Containers are lightweight and portable encapsulations of an environment in which to run applications.
- Containers are created from images. Inside a container, it has all the binaries and dependencies needed to run the application.

Registries and Repositories

- A registry is where we store our images.
- You can host your own registry, or you can use Docker's public registry which is called DockerHub.
- Inside a registry, images are stored in repositories.
- Docker repository is a collection of different docker images with the same name, that have different tags, each tag usually represents a different version of the image.





Docker Hub

A Public Docker Repository

Docker: concepts

Explore Help Search Sign up Log In

Explore Official Repositories

 nginx official	2.9K STARS	10M+ PULLS	> DETAILS
 busybox official	661 STARS	10M+ PULLS	> DETAILS
 ubuntu official	3.9K STARS	10M+ PULLS	> DETAILS
 registry official	813 STARS	10M+ PULLS	> DETAILS

<https://hub.docker.com/>

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

Deep Dive into Docker Containers

- **running containers in detached mode**
- **docker ps command**
- **docker container name**
- **docker inspect command**

Foreground vs Detached

	Run Container in Foreground	Run Container in Background
<i>Description</i>	Docker run starts the process in the container and attaches the console to the process's standard input, output, and standard error.	Containers started in detached mode and exit when the root process used to run the container exits.
<i>How to specify?</i>	default mode	-d option
<i>Can the console be used for other commands after the container is started up?</i>	No	Yes

Docker: interacting with containers

Playground

Use Docker in a safe playground environment

Install Docker

Installing docker to a VM

Our first container

Describes how to create your first container

Run a Webapp with Docker

Describes how to create your first container to run a webapp with Docker.

<https://katacoda.com/loodse/courses/docker>



Interacting with Containers

Learn how to interact with running container

Start Scenario



Docker Images and Layers

Understand layers in docker image

Start Scenario



Building Images Interactively

Learn how to build docker image interactively and tag them

Start Scenario



Building images with Dockerfile

Learn how to build docker image from Dockerfile

Start Scenario



Docker Build ignore



Naming and



Managing Log Files



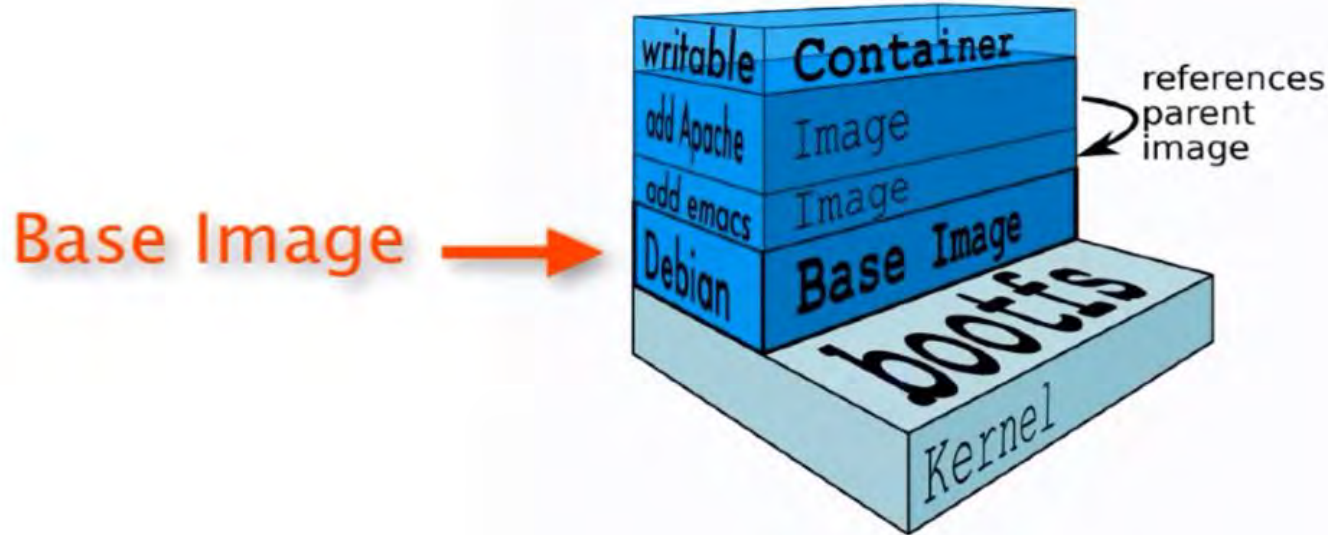
Networking

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

Docker Image Layers



Image Layers



Docker: image layers

Docker Images and Layers

Step 1 of 2

Step 1

Layers in docker image

Each Docker image references a list of read-only layers that represent filesystem differences.

Layers are stacked on top of each other to form a base for a container's root filesystem.

Pull `debian` image to your local system.

```
docker pull debian ✓
```

```
docker history debian ✓
```

 shows you list of layers that `debian` image contains.

More Information you can find out with:

```
docker inspect debian ✓
```

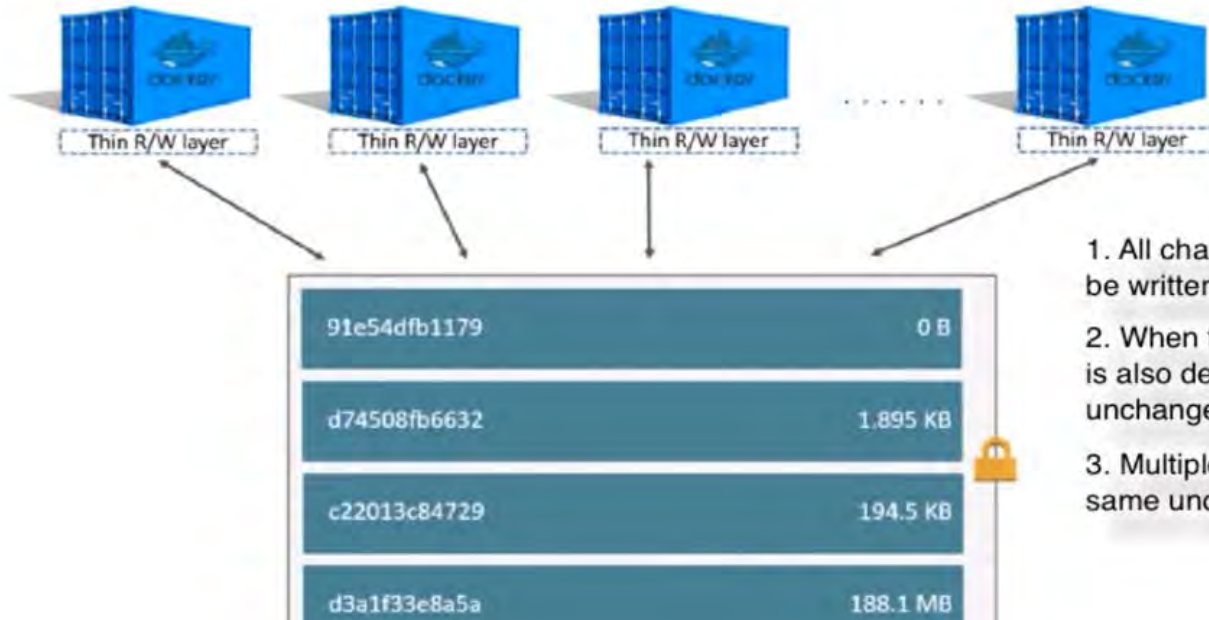
```
Terminal +
Your Interactive Bash Terminal.

$ docker pull debian
Using default tag: latest
latest: Pulling from library/debian
8f0fdd3eaac0: Pull complete
Digest: sha256:f19be6b8095d6ea46f5345e2651eec4e5ee9e84fc83f3bc3b73587197853dc9e
Status: Downloaded newer image for debian:latest
$ docker history debian
IMAGE                CREATED              CREATED BY
b5d2d9b1597b         3 weeks ago         /bin/sh -c #(nop)  CMD ["bash"]
<missing>            3 weeks ago         /bin/sh -c #(nop)  ADD file:d6d0bdf8cb07a7a0d...
$ docker inspect debian
[
  {
    "Id": "sha256:b5d2d9b1597bb7acf97f59b2927dce2645cf253ff804d5cf538c5ec82975769d
```

<https://katacoda.com/lloodse/courses/docker>

```
"debian@sha256:f19be6b8095d6ea46f5345e2651eec4e5ee9e84fc83f3bc3b7358719785
```

Image Layers



1. All changes made into the running containers will be written into the writable layer.
2. When the container is deleted, the writable layer is also deleted, but the underlying image remains unchanged.
3. Multiple containers can share access to the same underlying image.

Ways to Build a Docker Image

- Commit changes made in a Docker container.
- Write a Dockerfile.

Build Docker Images

Approach 1: committing changes made in a container

Docker: build image interactively

Building Images Interactively

Step 1 of 4

Step 1

Step 1 - Install Apache into running container

`docker commit` command creates an image from container's changes.

Start an Debian container:

`docker run -it debian` ✓ It will pull `debian` image to your local system if it does not exist already.

After pulling image, it will start container and open a shell into running container.

We'll install Apache into running container.

```
Terminal +
Your Interactive Bash Terminal.

$ docker run -it debian
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
8f0fdd3eaac0: Pull complete
Digest: sha256:f19be6b8095d6ea46f5345e2651eec4e5ee9e84fc83f3bc3b73587197853dc9e
Status: Downloaded newer image for debian:latest
root@ac64635bbebe:/#
```

<https://katacoda.com/loodse/courses/docker>

Steps

1. Spin up a container from a base image.
2. Install a package in the container.
3. Commit changes made in the container.

Docker commit

- Docker commit command would save the changes we made to the Docker container's file system to a new image.

```
docker commit container_ID repository_name:tag
```

Build Docker Images

Approach 2: Writing a Dockerfile

Dockerfile and Instructions

- A Dockerfile is a text document that contains all the instructions users provide to assemble an image.
- Each instruction will create a new image layer to the image.
- Instructions specify what to do when building the image.

Docker: building images with Dockerfile



Katacoda

Katacoda Overview & Solutions

Search

Claim Your Profile

Log Out

Building Images with Dockerfile

Step 1 of 6

Step 1

Write Dockerfile

In this step we shall create Dockerfile.

Create a Dockerfile with following content in current directory.

```
cat > Dockerfile << EOF
FROM ubuntu:18.04
RUN apt-get update && apt-get install apache2 -y && apt-get clean
CMD ["apache2ctl", "-DFOREGROUND"]
EOF ✓
```

Verify the file: `cat Dockerfile` ✓

Above Dockerfile content describes a `Debian jessie` Docker image with Apache web server installed.

- `FROM` indicates the base image for our build
- Each `RUN` line will be executed by Docker during the build
- Our `RUN` commands must be non-interactive. (You can't provide input to Docker during the build.)
- In many cases, we will add the `-y` flag to `apt-get`

📁

📄

📄

./root

├── .bashrc

├── .cache/

├── .hushlogin

├── .profile

├── .ssh/

├── Dockerfile

└── html/

Dockerfile

index.html

```
1 FROM ubuntu:18.04
2 RUN apt-get update && apt-get install apache2 -y && apt-get clean
3 CMD ["apache2ctl", "-DFOREGROUND"]
```

Terminal

+

Enabling site 000-default.
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Removing intermediate container 3d88bcc54407

<https://katacoda.com/lloodse/courses/docker>

Dockerize a Hello World Web Application



Docker: dockerize HTML website



Katacoda

Katacoda Overview & Solutions

Search

Claim Your Profile

Log Out

Deploy Static HTML Website as Container

Step 1 of 3

Step 1 - Create Dockerfile

Docker Images start from a base image. The base image should include the platform dependencies required by your application, for example, having the JVM or CLR installed.

This base image is defined as an instruction in the Dockerfile. Docker Images are built based on the contents of a Dockerfile. The Dockerfile is a list of instructions describing how to deploy your application.

In this example, our base image is the Alpine version of Nginx. This provides the configured web server on the Linux Alpine distribution.

Task

Create your *Dockerfile* for building your image by copying the contents below into the editor.

```
FROM nginx:alpine
COPY . /usr/share/nginx/html
```

Copy to Editor

The first line defines our base image. The second line copies the content of the current directory into the container.



Dockerfile
index.html

Dockerfile

```
1 FROM nginx:alpine
2 COPY . /usr/share/nginx/html
3
```

Terminal

docker:80



Your Interactive Bash Terminal. A safe place to learn and execute commands.

```
$
$ ls
Dockerfile index.html
$
$ cat index.html
```

<https://katacoda.com/courses/docker/>

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

MASTER KUBERNETES INTRODUCTION



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

WHAT & WHY - CONTAINERS, ORCHESTRATION



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

WHAT & WHY - CONTAINERS, ORCHESTRATION



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

WHAT & WHY - CONTAINERS, ORCHESTRATION

	CONTAINERS	CONTAINER ORCHESTRATION
FUNCTION	KEEP SOFTWARE SEPARATED INTO ITS OWN "CLEAN" VIEW OF AN OPERATING SYSTEM	DEFINE RELATIONSHIPS BETWEEN CONTAINERS, WHERE THEY COME FROM, HOW THEY SCALE, AND HOW THEY CONNECT TO THE WORLD AROUND THEM
PREDECESSORS/ ALTERNATIVES	<ul style="list-style-type: none">•VIRTUAL MACHINES•DIRECT INSTALLATION	<ul style="list-style-type: none">•HOMEGROWN SCRIPTS•MANUAL, BESPOKE STATIC CONFIGURATION BETWEEN CONTAINERS
PACKAGES/VENDORS	<ul style="list-style-type: none">•DOCKER•RKT•GARDEN•LXC•MESOS	<ul style="list-style-type: none">•KUBERNETES•DOCKER SWARM•AMAZON ECS•MESOS

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

THE CONTAINER "WAR" MAYBE COMING TO A CLOSE

- Docker is by-and-far the leader
- Defines:
 - A Container Format (Dockerfile)
 - Registry to host Containers (public and private)

THE ORCHESTRA(S) - ORCHESTRATION TECHNOLOGY

- A much newer space
- Kubernetes has quite a lead, and emerging as the clear leader & winner, it's closest competitor, Docker Swarm, recently announced support for Kubernetes-style configuration & compatibility
- Cooperation & formal standards bodies are more prevalent in the growth of orchestration

REVIEW & USING KUBERNETES ORCHESTRATING CONTAINERS

- Containers have helped simplify & standardize how software is modularized & deployed
- Container orchestration has arisen to help equally simplify & standardize how these containers come together to make a usable software system
- Kubernetes as a container orchestration standard

MASTER KUBERNETES

K8S OVERVIEW



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

KUBERNETES

- Born from a Google internal project in mid-2014 (Google "Borg")
- 1.0 Release in July 2015
- Google partnered with the Linux Foundation to form the Cloud Native Computing Foundation (CNCF) to offer Kubernetes as an open standard
- Frequently abbreviated "k8s" - Greek for "helmsman" or "pilot"

ADVANTAGES OF KUBERNETES

- Based on extensive experience from Google, over a long period of time.
- Large open source community & project, mature governing organization (CNCF)
- Auto-scaling,, cloud-agnostic-yet-integratable technologies

MAKING KUBERNETES YOURS

- Kubernetes runs anywhere Linux does
 - Your Laptop
 - Globally distributed data centers
 - Major cloud providers
 - Anywhere in between & nearly any combination (not that you'd want to necessarily)

K8S: overview



MASTER KUBERNETES

DEPLOYING K8S

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

WHERE DOES K8S LIVE?

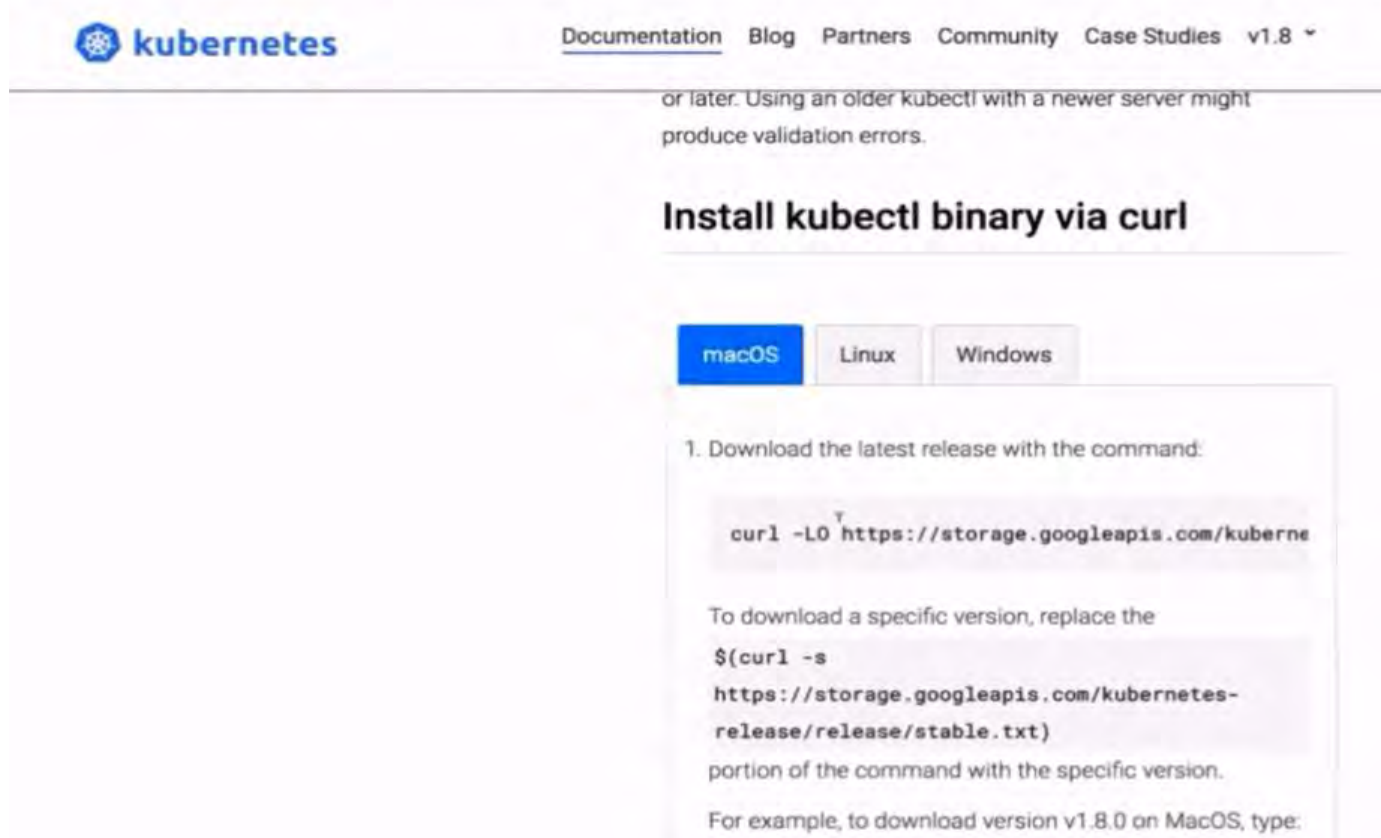
- Anywhere you need it to
- But, where would you want it to live?
- That depends on who you are & what you're using it for
 - Evaluation/Training - minikube (local version of k8s)
 - Development - minikube, dev cluster on a cloud provider
 - Deployment - cloud provider or bare metal

K8S ON YOUR WORKSTATION (MINIKUBE)

- minikube allows you to run the actual kubernetes code locally on your machine.
 - Avoid the complexity, expense, and slower response of a remote cluster
 - Well supported for development and testing
 - Not a production technology, defeats the purpose of container orchestration in many ways and cannot be relied upon for production workloads.

A FEW CONCEPTS TO NOTE

- Kubernetes "deployments" are the high-level construct that define an application
- "Pods" are instances of a container in a deployment
- "Services" are endpoints that export ports to the outside world
- You can create, delete, modify, and retrieve information about any of these using the *kubectl* command (as well as the Kubernetes local UI we will show later)

A screenshot of the Kubernetes documentation website. The top navigation bar includes links for Documentation, Blog, Partners, Community, Case Studies, and a version dropdown set to v1.8. The main content area shows a section titled 'Install kubectl binary via curl'. Below the title are three tabs: macOS (selected), Linux, and Windows. The macOS tab contains instructions for downloading the latest release using curl, followed by a code block showing the command: `curl -LO https://storage.googleapis.com/kubernetes-`. Below this, it explains how to download a specific version by replacing a portion of the command with a version number, such as v1.8.0. The text is partially obscured by a semi-transparent box.

or later. Using an older kubectl with a newer server might produce validation errors.

Install kubectl binary via curl

macOS Linux Windows

1. Download the latest release with the command:

```
curl -LO https://storage.googleapis.com/kubernetes-
```

To download a specific version, replace the `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` portion of the command with the specific version.

For example, to download version v1.8.0 on MacOS, type:

TESTING KUBECTL

```
kubectl version
```

```
overpriced-strip-of-glass:Downloads basitmustafa$ kubectl version
Client Version: version.Info{Major:"1", Minor:"8", GitVersion:"v1.8.2", GitCommit:"bdaeafa71f6c7c04636251031f93464384d54963", GitTreeState:"clean", BuildDate:"2017-10-24T21:08:42Z", GoVersion:"go1.9.1", Compiler:"gc", Platform:"darwin/amd64"}
Server Version: version.Info{Major:"1", Minor:"8+", GitVersion:"v1.8.1-gke.1", GitCommit:"aba494e68a76583d2d7d1b9c97e4a97a19c3a920", GitTreeState:"clean", BuildDate:"2017-10-27T23:54:39Z", GoVersion:"go1.8.3b4", Compiler:"gc", Platform:"linux/amd64"}
```

Playgrounds

Playgrounds give you a configured environment to start playing and exploring using an unstructured learning approach.

[SEE ALL PLAYGROUNDS](#)

Most Popular Playgrounds



Visual Studio
Code



Kubernetes
Playground



Docker Swarm
Playground



Ubuntu
Playground

<https://katacoda.com/courses/kubernetes/>

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

MASTER KUBERNETES

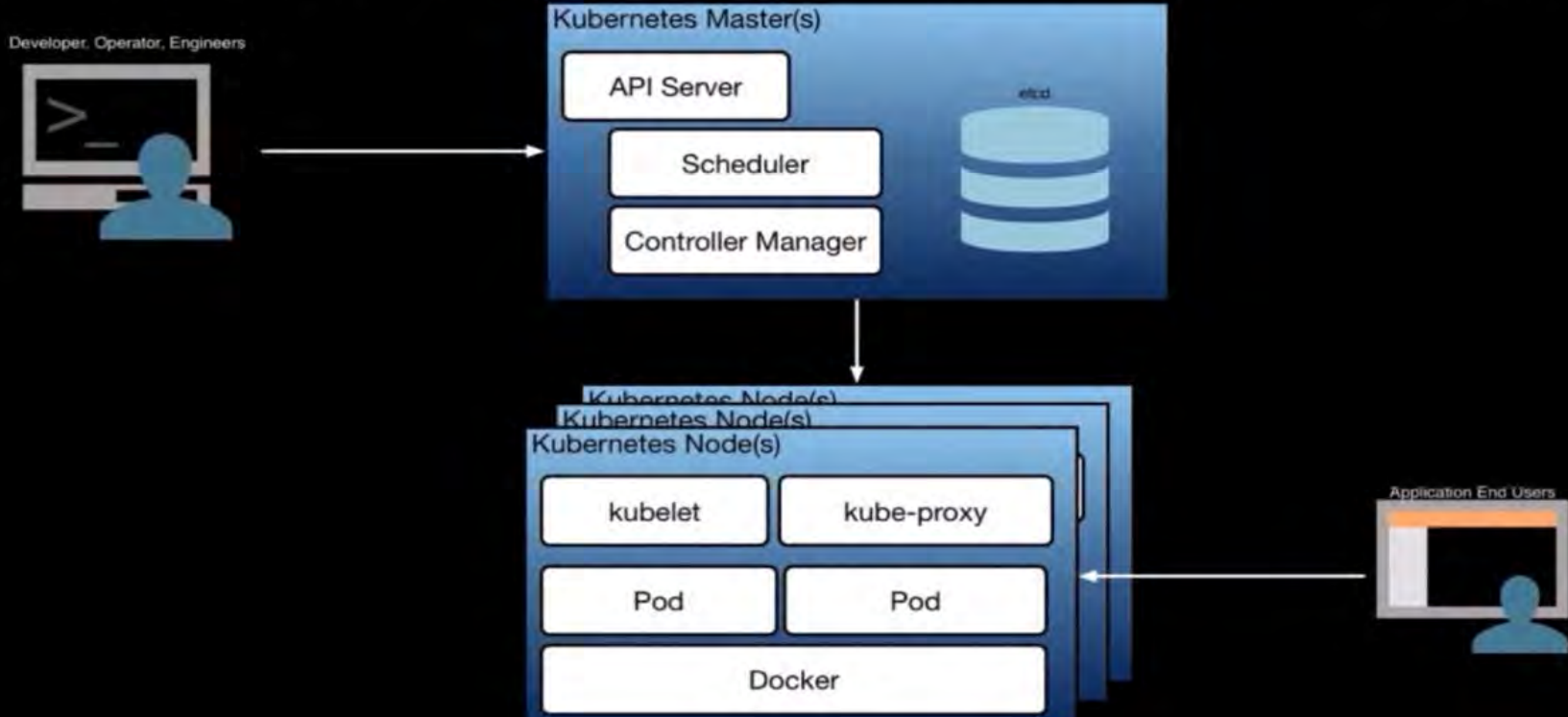
ARCHITECTURE OVERVIEW



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

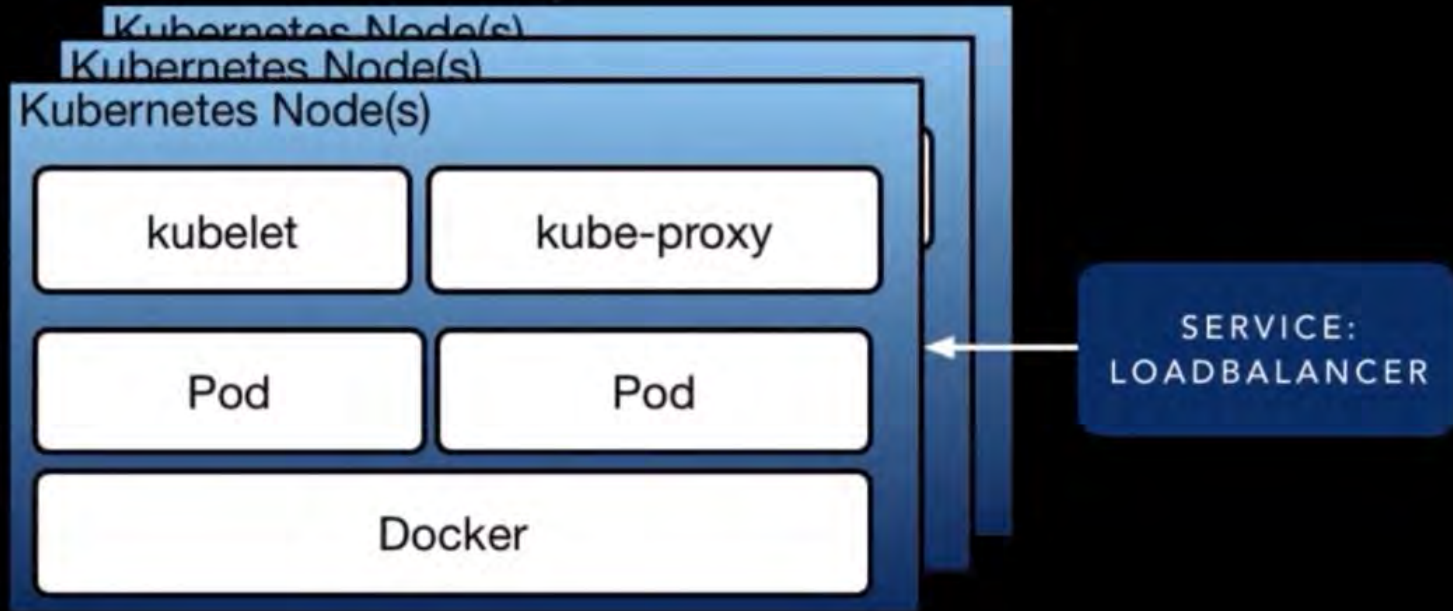
K8S: architecture overview

KUBERNETES FROM ABOVE (SIMPLIFIED)



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

A COMMON STARTING POINT: NODES



K8S: first app

Namespaces

Nodes

Persistent Volumes

Workloads

default

Deployments

Replication controllers

Daemon Sets

Pod Sets

Jobs

Pods

Services and discovery

Services

Ingress

Jobs

Name	Labels	Pods	Age
bootiful-couchbase	name: bootiful-couchbase-pod	0 / 1	1m

Replication controllers

Name	Labels
bootiful-couchbase-rc	app: couchbase-rc-pod

Pods

Name	Status	Restart	Age
bootiful-couchbase	Failed	0	54 seconds
bootiful-couchbase	Succeeded	0	35 seconds
couchbase-rc-32...	Running	0	a minute

MASTER KUBERNETES
YOUR FIRST K8S APP

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

WHAT DOES A K8S "APP" LOOK LIKE?

- "Deployments" are the central metaphor for what we'd consider "apps" or "services"
- Deployments are described as a collection of resources and references
- Deployments take many forms based on the type of services being deployed
- Typically described in YAML format

PRACTICAL: A TOMCAT DEPLOYMENT

- We'll deploy the Tomcat App Server using the official docker image
- Key Tasks:
 - Define the deployment
 - Expose its services
 - Deploy it to our cluster

Kubernetes Fundamentals

By loodse

Learn how to run scale and manage containers with Kubernetes

<https://katacoda.com/loodse/courses/kubernetes>



Kubernetes Playground and Editor

Loodse Kubernetes Playground with Editor in a safe environment.



Kubernetes Playground and 2 Terminals

Loodse Kubernetes Playground with 2 Terminals in a safe environment.



Setup a kubernetes cluster with kubeadm

Use kubeadm to setup a kubernetes cluster



Deploy a simple application

Deploy a simple application in a kubernetes cluster

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

PRACTICAL: A TOMCAT DEPLOYMENT

deployment.yaml

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: tomcat-deployment
spec:
  selector:
    matchLabels:
      app: tomcat
  replicas: 1
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
        - name: tomcat
          image: tomcat:9.0
          ports:
            - containerPort: 8080
```

K8S: basic kubectl

Namespaces

Nodes

Persistent Volumes

default

MASTERS KUBERNETES

Deployments

Replication Controllers

Daemon Sets

Pod Sets

Jobs

Pods

Services and discovery

Services

Ingress

Jobs

Name	Labels	Pods	Age
 beautiful-couchbase	name: beautiful-couchbase-pod	0 / 1	

Replication controllers

Name	Labels
app: couchbase-ro-pod	

Pods

Name	Status	Reasons	Age
 beautiful-couchba...	Failed	0	54 seconds
 beautiful-couchba...	Succeeded	0	35 seconds
 couchbase-ro-2z...	Running	0	a minute



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

KUBECTL GET PODS

```
$ kubectl get pods
```

- Lists all pods in all namespaces
- Provides the pod name, how many instances of the pod are running & ready, its status, how many times they have restarted, and their age

Sample Output

```
root@vt-es-1:~# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
tomcat-deployment-7bd7889564-717n2 1/1     Running   0           15m
root@vt-es-1:~# █
```

```
$ kubectl get pods [pod name]
```

- ## Sample Output

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

KUBECTL EXPOSE PORT

```
$ kubectl expose <type name> <identifier/name> [--port=external port] [--target-port=container-port] [--type=service-type]
```

- Exposes a port (TCP or UDP) for a given deployment, pod, or other resource

Sample Output

```
[root@vt-es-1:~# kubectl expose deployment tomcat-deployment --type=NodePort  
service "tomcat-deployment" exposed
```

KUBECTL EXEC

```
$ kubectl exec [-it] <pod name> [-c CONTAINER] -- COMMAND [args...]
```

- Execute a command in a container
- -i option will pass stdin to the container
- -t option will specify stdin is a TTY

Sample Output

```
root@vt-es-1:~# kubectl exec -it tomcat-deployment-7bd7889564-717n2 bash
root@tomcat-deployment-7bd7889564-717n2:/usr/local/tomcat# whoami
root
root@tomcat-deployment-7bd7889564-717n2:/usr/local/tomcat# █
```

K8S: scaling & replication

MASTER KUBERNETES

SCALING & REPLICATION



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

SCALING APPLICATIONS

- Stateful vs Stateless Applications
- How state is handled can either enable or interfere with scaling, it most certainly dictates how you'll scale

REPLICAS

- Kubernetes allows you to define replicas when you deploy an application - you have a few options
 - Setting "replica" in your Deployment (recommended)
 - Defining a ReplicaSet
 - Bare Pods
 - Job
 - DaemonSet

SCALING - A PRACTICAL EXAMPLE

deployment.yaml

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: tomcat-deployment
spec:
  selector:
    matchLabels:
      app: tomcat
  replicas: 4
  template:
    metadata:
      labels:
        app: tomcat
    spec:
      containers:
        - name: tomcat
```

<https://katacoda.com/loodse/courses/kubernetes>

SCALING - A PRACTICAL EXAMPLE

```
$ kubectl scale --replicas=4 deployment/tomcat-deployment
```

<https://katacoda.com/lloodse/courses/kubernetes>

K8S: deployments

MASTER KUBERNETES

DEPLOYMENTS



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

WHAT ARE DEPLOYMENTS?

- Deployments are a high-level object in Kubernetes, they define a desired state of an application
- They consist of Pods
- Optionally, they can also include ReplicaSets (that are automatically managed by the Deployment based on its replica configuration)

WITH DEPLOYMENT OBJECTS YOU CAN:

- Create a new deployment
- Update an existing deployment
- Apply rolling updates to Pods running on your cluster
- Rollback to a previous version
- Pause & Resume a deployment

K8S: web interface

MASTER KUBERNETES

WEB INTERFACE



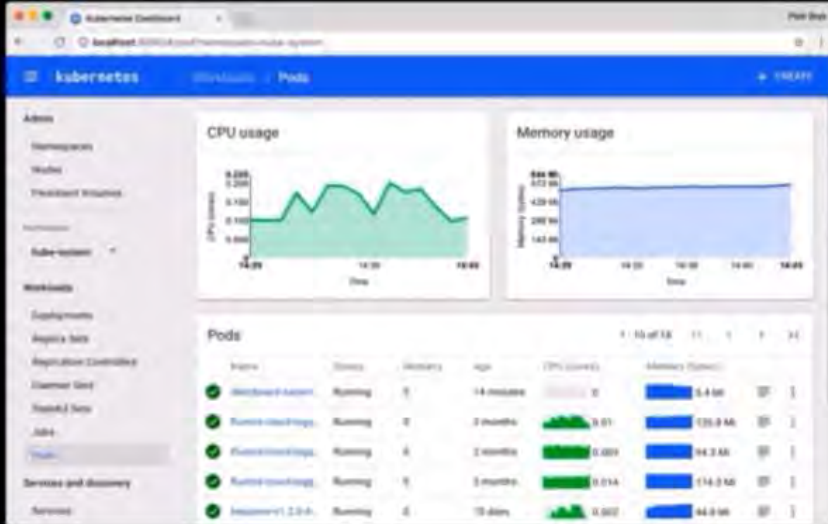
Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

KUBERNETES WEB UI

- kubectl is great, but sometimes a web UI is useful, too
 - Called the “Dashboard UI”
- Runs on your Kubernetes master(s)
- Accessible directly if you have direct network connectivity directly to your cluster/master(s) (unlikely in production situations)
 - kubectl can create a proxy/tunnel for you in situations you do not

```
$ kubectl proxy
```

KUBERNETES WEB UI



- Provides a variety of views for nearly anything in your Kubernetes cluster
- Allows update, deletion, and creation of nearly anything in your Kubernetes cluster
- Accesses the same APIs as kubectl

USING THE WEB UI

- On some Kubernetes clusters the Dashboard UI is pre-installed, on some it is not (many cloud providers' Kubernetes services include it)
- Installing the Dashboard

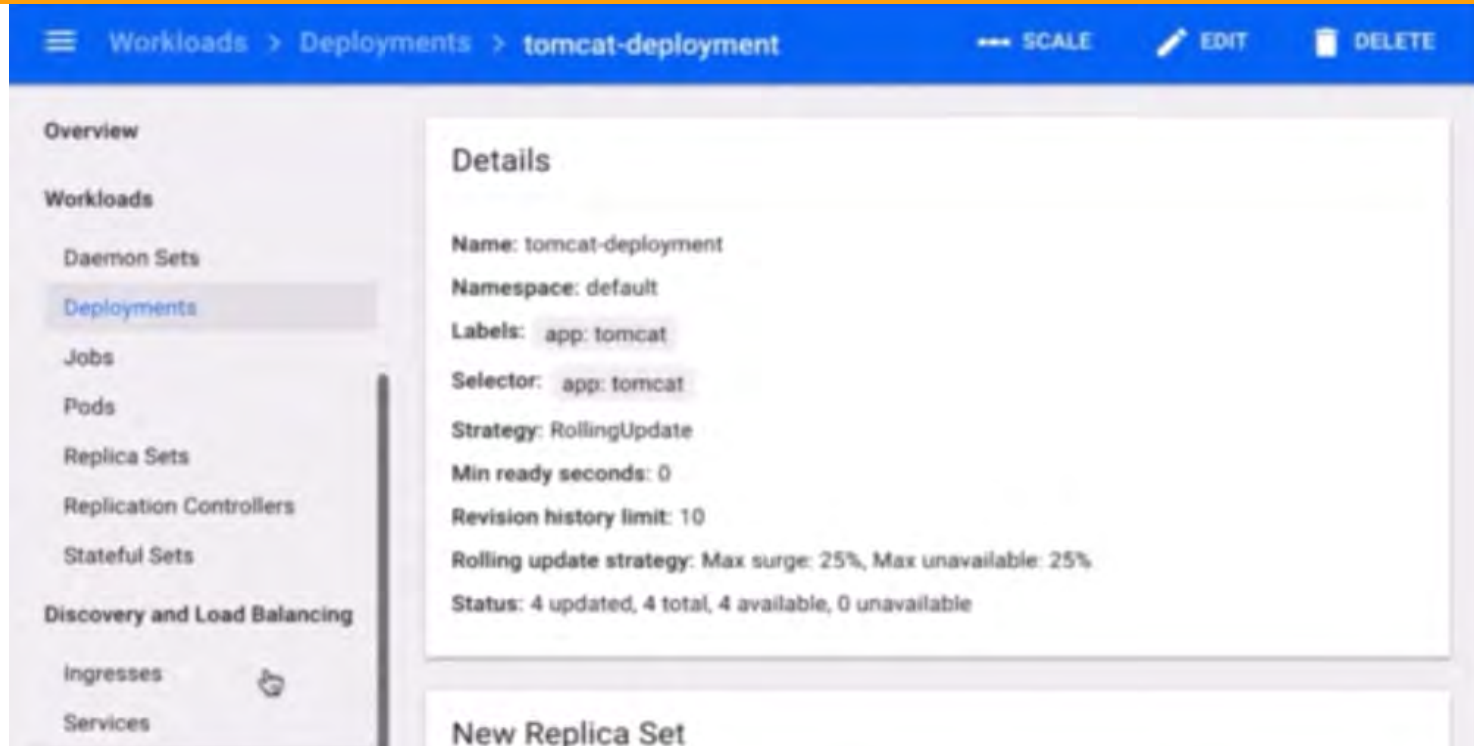
```
$ kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml
```

- Accessing the dashboard (the most fool-proof way)

```
$ kubectl proxy
```

- Navigate to "<http://localhost:8001/ui>" in your web browser

K8S: web interface



<https://www.katacoda.com/mjboxboat/courses/kubernetes-basic/>

MASTER KUBERNETES

EXERCISE: DEPLOYING & SCALING



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

WHAT YOU'VE LEARNED

- How to define a deployment in Kubernetes
- How to deploy this deployment to Kubernetes
- How to scale this deployment in Kubernetes

YOUR TASK

- Use what you've learned to deploy and scale MongoDB (a popular NoSQL datastore)
- Success criteria:
 - The current version of MongoDB is running on your Kubernetes cluster with four replicas
- A few hints:
 - MongoDB listens on port 27017
 - Officially supported MongoDB Docker images are available on Docker Hub

A POSSIBLE SOLUTION

- Use kubectl run to run the default mongo image

```
$ kubectl run mongo-exercise-1 --image=mongo --port=27017
```

- Use kubectl scale to scale the deployment to 4 replicas

```
$ kubectl scale --replicas=4 deployment/mongo-exercise-1
```

OTHER POSSIBLE SOLUTIONS

- Write a deployment.yaml file, use kubectl apply, & use kubectl expose to expose a service
- Write a deployment.yaml file & a service.yaml file and use kubectl apply on both
- Use a Kubernetes package manager like helm to handle the work for you

Helm: fundamentals



HELM FUNDAMENTALS

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

HELM FUNDAMENTALS

- What Is Helm?
- Helm Installation Guides for all Operating Systems
- First Helm Deployment
- Helm Charts and First Chart Creation
- Using Helm Template Calls
- Helm Value Files
- Common Helm Commands

WHAT IS HELM?

- Kubernetes can become incredibly complex with all the objects you need to manage
- Helm provides a simple way to package and configure exactly what you need by combining everything into one easy deployment
- Helm fills the need to be able to quickly and reliably provision container applications through easy installation, update, and removal

WHAT IS HELM?



- With Helm and its 'charts', you can define, run, and upgrade unique and complex Kubernetes apps
- The package manager helps developers streamline the installation and management of Kubernetes applications
- Helm Charts are deployment packages all bundled up

HOW DOES HELM WORK?

- The package manager is comprised of two parts:
 - The tool itself (the 'helm')
 - The server (the 'tiller') which runs inside Kubernetes clusters
- When you input the Helm install command, a Tiller Server receives the request and installs the appropriate package

HELM CHARTS - OFFICIAL AND UNOFFICIAL

- Helm Charts gives you the ability to leverage Kubernetes packages through the click of a button or single CLI command
- Using the 'helm' tool, you can locate, customize, utilize, and install any of these charts
- All Helm Charts contain at least two things:
 - A package description (chart.yaml)
 - One or more templates, which contain Kubernetes manifest files

ACCESSING HELM CHARTS

- Helm offers a huge repository of both official and unofficial charts on GitHub
- Charts can be stored on a disk or just pull them from remote chart repositories as needed
- To install a chart, use 'helm install stable/<chart>'

HELM REVIEW

- Helm provides a simple way to programmatically package everything into one simple deployment
- With Helm and its 'charts', developers can quickly define, run, and upgrade even really complex Kubernetes apps
- Helm offers a huge repository of both official and unofficial charts on GitHub
- Charts can be stored on a disk or just pull them from remote chart repositories as needed

Helm: installation



MASTER KUBERNETES

HELM INSTALLATION GUIDES



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

HELM INSTALLATION GUIDES

- There are two parts to install for Helm: The 'helm' client and the 'tiller' server
- Options for 'helm' client installation include:
 - Binary releases
 - From Snap (for Linux)
 - From Homebrew (for macOS)
 - From Chocolatey (for Windows)

- Fixed the `--tiller-namespace` flag for helm plugins
- Suppressed the 'unauthenticated users' warning when `--tiller-tls-verify` is present on `helm init`

There were so many fixes this release that we're probably missing a few noteworthy ones, so we suggest by having a look at the changelog for the full list of fixes and enhancements!

Installation and Upgrading

Download Helm 2.10. The common platform binaries are here:

- [MacOS amd64 \(checksum\)](#)
- [Linux amd64 \(checksum\)](#)
- [Linux arm \(checksum\)](#)
- [Linux arm64 \(checksum\)](#)
- [Linux i386 \(checksum\)](#)
- [Linux ppc64le \(checksum\)](#)
- [Windows amd64 \(checksum\)](#)

Once you have the client installed, upgrade Tiller with `helm init --upgrade`.

<https://github.com/helm/helm/releases>

HELM INSTALLATION GUIDES - TILLER SERVER

- Connect to your kube cluster
- Install the 'tiller' server into the cluster by running `helm init`
 - This checks the local environment is set up right and connects to a default cluster
 - Once connected, it will install 'tiller' into the `kube-system` namespace
 - You should now be able to run `kubectl get pods --namespace kube-system` to see 'tiller' up and running

Helm: first deployment

MASTER KUBERNETES

FIRST HELM DEPLOYMENT



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

Helm: first deployment

Helm Package Manager

Step 1 of 4 ▶

Install Helm

Helm is a single binary that manages deploying Charts to Kubernetes. A chart is a packaged unit of kubernetes software. It can be downloaded from

<https://github.com/kubernetes/helm/releases>

```
curl -LO https://storage.googleapis.com/kubernetes-helm/helm-v2.8.2-linux-amd64.tar.gz
tar -xvf helm-v2.8.2-linux-amd64.tar.gz
mv linux-amd64/helm /usr/local/bin/ ↵
```

Once installed, initialise update the local cache to sync the latest available packages with the environment.

```
helm init
helm repo update ✓
```

Terminal



Your Interactive Hands On Lab Terminal

```
master $ launch.sh
Waiting for Kubernetes to start...
Kubernetes started
master $ helm init
Creating /root/.helm
Creating /root/.helm/repository
Creating /root/.helm/repository/cache
Creating /root/.helm/repository/local
Creating /root/.helm/plugins
Creating /root/.helm/starters
Creating /root/.helm/cache/archive
Creating /root/.helm/repository/repositories.yaml
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
Adding local repo with URL: http://127.0.0.1:8879/charts
$HELM_HOME has been configured at /root/.helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes Cluster.

Please note: by default, Tiller is deployed with an insecure 'allow unauthenticated users' policy.
To prevent this, run `helm init` with the --tiller-tls-verify flag.
For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation
```

<https://katacoda.com/courses/kubernetes/helm-package-manager>



MASTER KUBERNETES

HELM CHARTS & FIRST CHART CREATION

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

HELM CHARTS INTRODUCTION

- As mentioned previously, there's both an official and unofficial repository
- These contain prepackaged charts for popular open-source software projects
- The charts each consist of a few YAML configuration files and some templates that are rendered into Kubernetes manifest files

BASIC HELM CHARTS DIRECTORY STRUCTURE

- Here is the basic directory structure of a chart:

```
package-name/  
  charts/  
  templates/  
  Chart.yaml  
  LICENSE  
  README.md  
  requirements.yaml  
  values.yaml
```

HELM CHARTS INTRODUCTION

- The `helm` command can install a chart from a local directory, or from a `.tar.gz` packaged version of this directory structure
- Packaged charts can be automatically downloaded and installed from chart repositories or repos

HELM CHART CREATION

- Use the `helm create` command to scaffold out a chart example we can build on
- The following command will create a new chart named `mychart` in a new directory:
`$ helm create mychart`

HELM CHART CREATION

- The result will look like this:

```
mychart
|-- Chart.yaml
|-- charts
|-- templates
|   |-- NOTES.txt
|   |-- _helpers.tpl
|   |-- deployment.yaml
|   |-- ingress.yaml
|   |-- service.yaml
-- values.yaml
```

- The most important element to note here is the templates/ directory
- This is where Helm finds the YAML definitions for your Services, Deployments, and other Kubernetes objects

HELM CHARTS & FIRST CHART CREATION - REVIEW

- Helm packages are called charts, and they consist of a few YAML configuration files and some templates that are rendered into Kubernetes manifest files
- The `helm` command can install a chart from a local directory, or from a `.tar.gz` packaged version of this directory structure
- A chart usually comes with default configuration values in its `values.yaml` file
- Helm can output the scaffold of a chart directory with `helm create chart-name`



MASTER KUBERNETES

USING HELM TEMPLATE CALLS

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

USING HELM TEMPLATE CALLS

- When 'tiller' evaluates a chart, it sends all files in the templates/ directory through the template rendering engine
- Templates, when combined with values, will generate valid Kubernetes manifest files
- The Chart.yaml file contains a chart description which is accessible from within a template
- A template directive is enclosed in {{ and }} blocks

SIMPLE TEMPLATE CALL

- The template directive `{{ .Release.Name }}` will insert a release name into the YAML template
- Think of values that are injected into a template as namespaced objects, where a dot (.) separates each element
- The leading dot before 'Release' shows that we start with the top-most namespace
- `{{ .Release.Name }}` reads then as "start at the top namespace, find the Release object, then look inside it for an object called Name"

TEST TEMPLATE RENDERING

- To test the template rendering, but not actually install anything, you can use `helm install --debug --dry-run ./mychart`
- This will send the chart to the 'tiller server,' which will create the templates
- Instead of installing the chart though, it will return the rendered template so you can see the output

Helm: value files



MASTER KUBERNETES

HELM VALUE FILES

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

HELM VALUE FILES

- Values is by far the most important built-in object in the templates directory
- It gives you access to all the values configured in the values.yaml of your chart, it's sub-charts, and any values files or values provided directly on the command line
- Values files are plain YAML files

HELM VALUE FILES

- Using incorrect values in config files or failing to roll out apps correctly from YAML templates can break deployments
- By preconfiguring certain values through Helm Charts and setting others to sensible defaults, you create a consistent interface for a changing configuration
- This dramatically reduces complexity, and eliminates deployment errors by locking out incorrect configurations

PRE-INSTALLATION CONFIGURATION

- You can customize a chart before installing it by:
 - Running `helm inspect values`
 - Seeing the current configuration
 - And then overriding it during installation

VALUE FILES CONFIGURATION

- To provide values for a template in a specific chart:
 - Use a file called `values.yaml` inside all created charts, which contains default values
 - OR supply a YAML file that contains values (this can be provided in the `helm install` command)
 - Using `-f, --values valueFiles` flags allows you to specify values in a YAML file or a URL (can specify multiple) (default [])

ACCESSING VALUES

- Go templates provide the typical control structures you'll find in nearly all templating languages: `if / else`, and `range` (loop)
- Example:

```
{{- if .Values.deployment.volumes }}  
volumes:  
  {{- range .Values.deployment.volumes }}  
  - name: {{ .name }}  
    secret:  
      secretName: {{ .secretName }}  
  {{- end }}  
{{- end }}
```

PREDEFINED VALUE FILES

- There are several predefined values which cannot be overridden, these are:

- `Release.Name`
- `Release.Time`
- `Release.Namespace`

HELM VALUE FILES - REVIEW

- Values is by far the most important built-in object in the templates directory
- It gives you access to all the values configured in the values.yaml of your chart
- Using incorrect values in config files or failing to roll out apps correctly from YAML templates can break deployments
- You can customize a chart before installing it by running `helm inspect values`, seeing the current configuration, and then overriding it during installation
- There are several predefined values which cannot be overridden, these are `Release.Name`, `Release.Time`, & `Release.Namespace`



MASTER KUBERNETES

COMMON HELM COMMANDS

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

HELM CREATE

- `helm create` - Build and name a new chart
- Use the command to create a chart directory as well as the necessary common files and directories the chart will use
- E.g., `helm create foo` will create the following directory structure:

```
foo/
├── .helmignore # Contains patterns to ignore when packaging Helm charts.
├── Chart.yaml # Information about your chart
├── values.yaml # The default values for your templates
├── charts/ # Charts that this chart depends on
└── templates/ # The template files
```

HELM FETCH

- `helm fetch` - Download and unpack a chart from a repository into a local directory
 - Useful for fetching packages to examine, adjust, or repackage
 - Will also run a cryptographic verification of a chart without installing it
 - If a `-verify` flag is named, the requested chart **MUST** have a provenance file, and **MUST** pass the verification analysis
 - Use `helm fetch [flags] [chart URL | repo/chartname] [...]`

HELM UPGRADE

- `helm upgrade` - Upgrade a requested release
 - Specify 'release' and 'chart' arguments
 - The chart argument can be one of:
 - i. A chart reference('stable/mariadb')—use `--version` and `--devel` flags for older versions
 - ii. A chart directory path
 - iii. A packaged chart
 - iv. A full URL
 - Use `helm upgrade [RELEASE] [CHART] [flags]`

HELM STATUS

- `helm status` - Shows the status of the requested release
 - The status will include:
 - i. Last deployment time
 - ii. Kube namespace where the release lives
 - iii. Release state (UNKNOWN, DEPLOYED, DELETED, SUPERSEDED, FAILED or DELETING)
 - iv. List of resources that this release consists of, sorted by kind
 - v. Details on last test suite run (if applicable)
 - vi. Additional notes from the chart
- Use `helm status [flags] RELEASE_NAME`

HELM ROLLBACK

- `helm rollback` - Allows you to roll back a release to a previous version
 - The rollback command's first argument is the 'name' of a release
 - The second is the 'revision' (version) number
 - To display revision numbers, run `helm history RELEASE`
 - Use `helm rollback [flags] [RELEASE] [REVISION]`

KubeInvaders: a gamified chaos engineering tool

KubeInvaders is a gamified chaos engineering tool for Kubernetes. It is like Space Invaders but the aliens are PODs



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

KubeInvaders: a gamified chaos engineering tool

A screenshot of the GitHub repository page for 'lucky-sideburn / KubeInvaders'. The page shows the repository name, a navigation bar with links to Code, Issues (2), Pull requests (0), Projects (0), Security, and Insights. A 'Join GitHub today' banner is present. Below the banner, the repository is described as 'Chaos Engineering Tool for Kubernetes and Openshift' with tags for 'chaos', 'kubernetes', and 'openshift'. A progress bar shows 172 commits, 5 branches, 0 packages, 6 releases, 6 contributors, and Apache-2.0 license. At the bottom, there is a table of files: 'assets' (bundle.html5, 9 months ago), 'doc' (first commit, 10 months ago), and 'helm-charts' (Add missing 'labels' keyword to secrets.yaml, 2 months ago).

lucky-sideburn / KubeInvaders

Watch 5 Star 385 Fork 31

Code Issues 2 Pull requests 0 Projects 0 Security Insights

Join GitHub today

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Chaos Engineering Tool for Kubernetes and Openshift

chaos kubernetes openshift

172 commits 5 branches 0 packages 6 releases 6 contributors Apache-2.0

Branch: master New pull request Find file Clone or download

lucky-sideburn Merge pull request #17 from Avtandilko/patch-1 Latest commit 4ba5676 on 29 Nov 2019

assets	bundle.html5	9 months ago
doc	first commit	10 months ago
helm-charts	Add missing "labels" keyword to secrets.yaml	2 months ago

<https://github.com/lucky-sideburn/KubeInvaders>

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

KubeInvaders: a gamified chaos engineering tool



Katacoda Overview & Solutions

Search

Your Profile

Log Out

Helm Package Manager

Step 1 of 4 ▶

Terminal



Your Interactive Hands On Lab Terminal



```
helm search nginx
helm install --name nginx stable/nginx-ingress
kubectl get services

git clone https://github.com/lucky-sideburn/KubeInvaders.git
vi helm-charts/kubeinvaders/values.yaml
helm upgrade --install kubeinvaders --recreate-pods --namespace kubeinvaders ./helm-charts/kubeinvaders

watch -n .5 kubectl get pods,deploy,svc -o wide -n kubeinvaders
```

To prevent this, run `helm init` with the `--tiller-tls-verify` flag.

For more information on securing your installation see: https://docs.helm.sh/using_helm/#securing-your-helm-installation

<https://katacoda.com/courses/kubernetes/helm-package-manager>

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond



kubernetes

Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond



Passionate thinkers with the tiny little obsession of making business scalable and profitable, in energy and beyond

- <https://katacoda.com>
- <https://www.level-up.one>
- <https://github.com/lucky-sideburn/KubeInvaders>

The background of the slide is a deep blue space scene. On the right side, a large, curved portion of the Earth is visible, showing cloud patterns and the horizon. The rest of the background is filled with a dense field of stars and a faint, hazy band of light, possibly representing a nebula or the Milky Way.

THANK YOU

[Baxenergy.com](https://baxenergy.com)

This document is proprietary to BaxEnergy Italia Srl and contains reserved information and trade secrets: it is supplied in confidence and it is intended solely for the use of the individual entity to whom it is addressed. It should not be disclosed, duplicated, translated or otherwise revealed in whole or in part without the prior written consent of BaxEnergy Italia Srl. All rights are reserved.
Copyright© 2019 BaxEnergy Italia S.r.l. All rights reserved. EnergyStudioPro®, AssetStack and their module are registered trademarks of BaxEnergy Italia S.r.l.