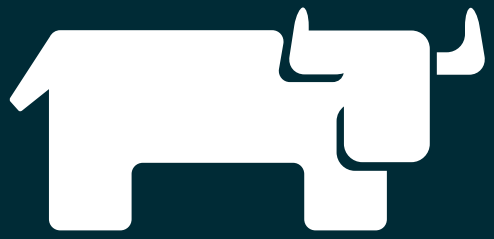
RANCHER RODEO 2.X



RANCHER®

INTRODUCTION

OBJECTIVES FOR THE DAY

- Explain Docker and Kubernetes Concepts
- Provision Machines
- Install Rancher Server
- Deploy a Kubernetes cluster
- Deploy Wordpress into the cluster

IF WE HAVE ENOUGH TIME...

- Persistent Storage
- Monitoring with Prometheus
- Using the Rancher CLI

PREREQUISITES

MINIMUM REQUIREMENTS

- Basic working knowledge of Docker
- Laptop with network card or WiFi to connect to the Internet
- A local copy of https://github.com/rancher/rodeo/ on your computer.

EXTRA CREDIT

- Kubectl installed for your platform:
  - https://kubernetes.io/docs/tasks/tools/install-kubectl/

IF YOU'RE USING VAGRANT

MINIMUM REQUIREMENTS

- You have a machine with enough resources for:
  - 1 core / 1.5G of RAM for the Rancher server
  - 2 cores / 3G of RAM per node (only one node is required)
- Vagrant is already installed
  - https://www.vagrantup.com/
- You've successfully deployed a VM with Vagrant
  - https://www.vagrantup.com/intro/getting-started/
- If using Windows, install OpenSSH for Windows
  - https://www.mls-software.com/opensshd.html

PREFERRED CONDITIONS

- You've already run `vagrant up` from https://github.com/rancher/rodeo/

IF YOU'RE USING A CLOUD PROVIDER

MINIMUM REQUIREMENTS

- You have access, permissions, and familiarity with launching instances in your preferred cloud provider

PREFERRED CONDITIONS

- Your machines are up and running with Docker installed (versions from 1.12.3 to 17.03.x are supported)
- Terraform scripts for AWS and Digital Ocean are available at https://github.com/rancher/rodeo

IS EVERYONE READY?

DOCKER OVERVIEW

WHAT'S A CONTAINER?

A container is a way to package application code and all of its runtime dependencies into a single, portable unit.

WHAT?

WHAT'S A CONTAINER?

A developer writes some code.

This code and all of its dependencies are built into a container image.

That container image can be run anywhere.

Containers spawned from that image are guaranteed to always be the same.

Containers run in an isolated namespace on the host, only aware of themselves and their processes.

CONTAINERS ARE NOT VMS

Containers share the host kernel. VMs utilize a hypervisor to virtualize the host hardware.

Containers do specific tasks. VMs do generic tasks.

Containers execute processes at near native speed.

Containers are immutable and portable.

WHY USE A CONTAINER?

When you use a container...

The underlying host doesn't need to manage dependencies. It only needs to host the engine that runs containers.

You have a clean, safe, hygenic runtime environment for your application.

You build your application once, and you run it anywhere.

You can scale your application horizontally by spawning more containers. Each container is guaranteed to be identical.

You can automate testing, integration, packaging, and anything else that you can script.

You can upgrade with no downtime.

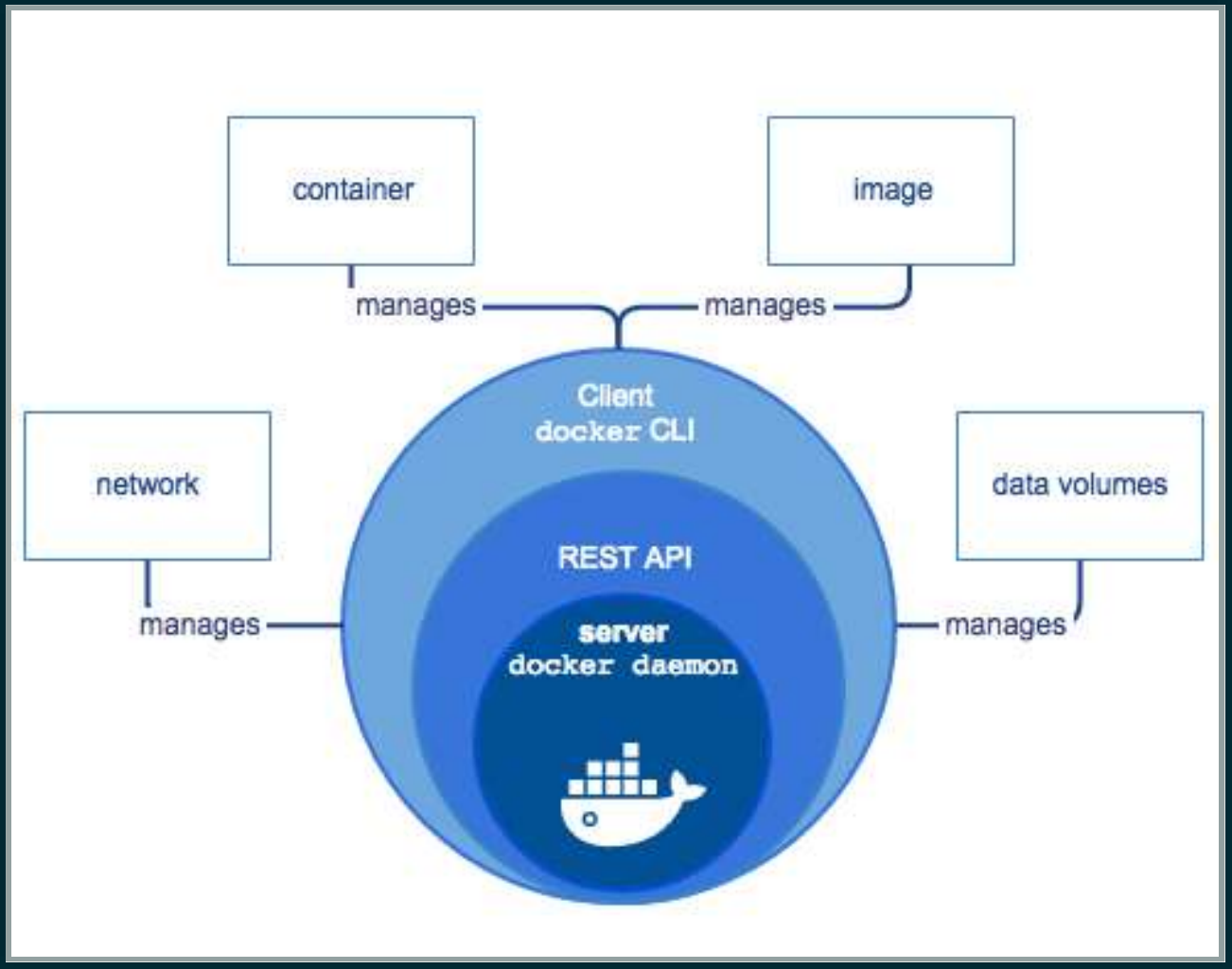
WHAT'S DOCKER?

Docker is a platform used to develop, deploy, and execute applications that run inside of containers.

The Docker Engine spawns containers from container images and acts as a bridge between host resources and those containers.

Docker containers are at the core of Kubernetes.

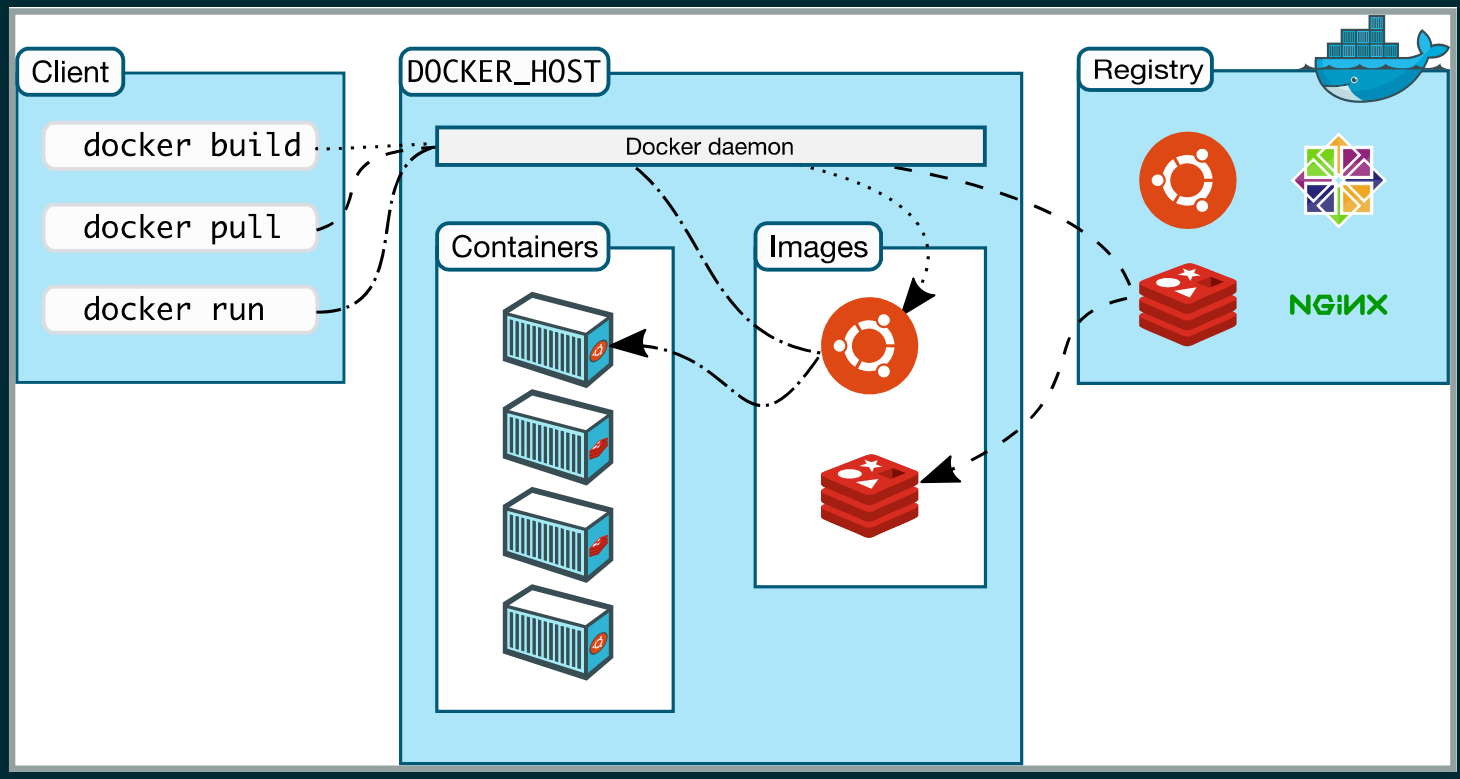
DOCKER ARCHITECTURE



- Server
- REST API
- Client CLI

DOCKER OBJECTS

- Registry
- Image
- Container



REVIEW

What's the purpose of a container registry?

> *A container registry holds container images.*

What's a container image?

> *A container image is a template that defines how to create a container.*

What's a container?

> *A container is a runnable instance of an image.*

KUBERNETES OVERVIEW

WHAT CAN WE SAY ABOUT KUBERNETES?

Kubernetes...

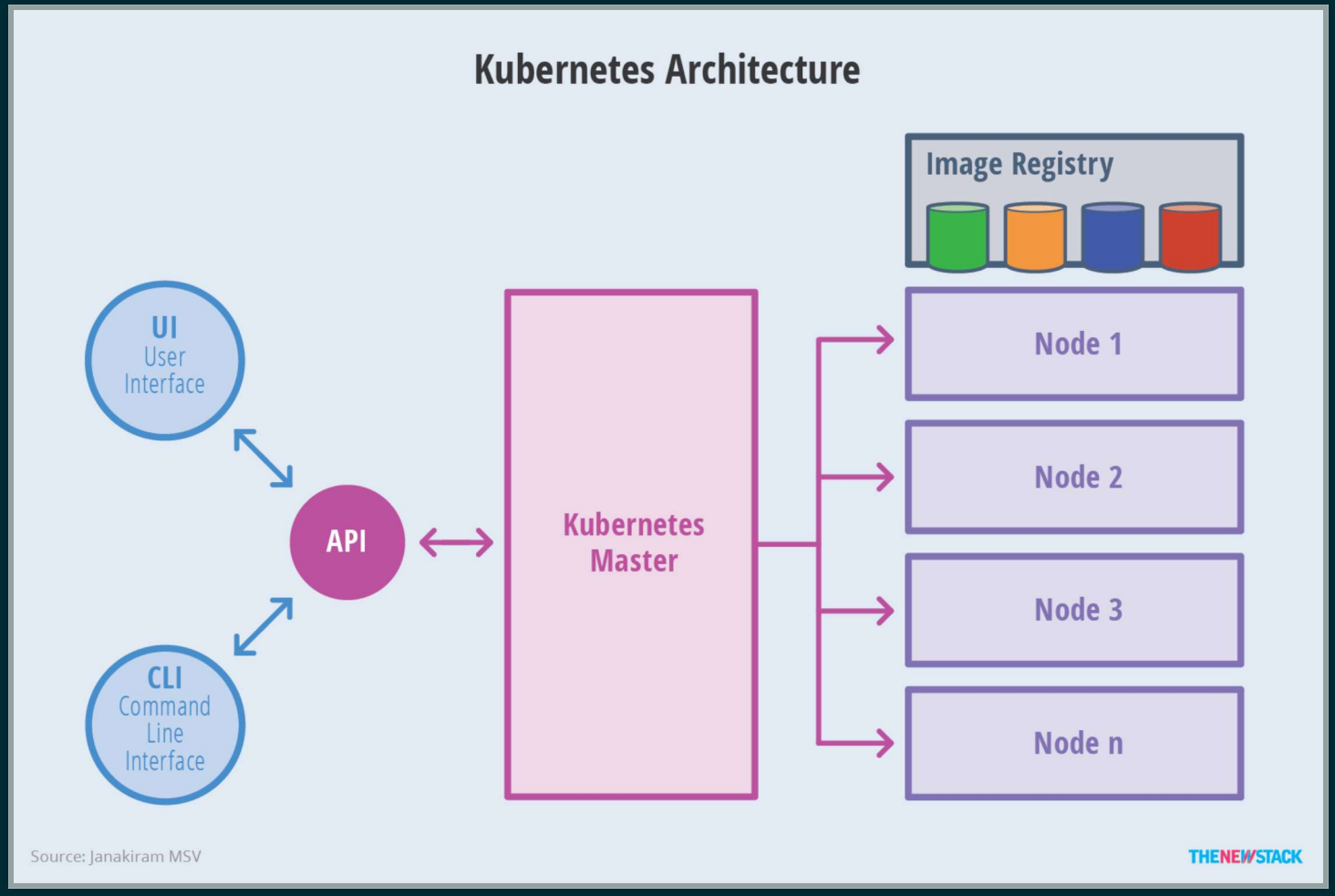
...is a powerful open-source platform for managing containerized workloads and services.

...uses a declarative syntax to describe the desired state of the cluster.

...has a large, rapidly growing ecosystem.

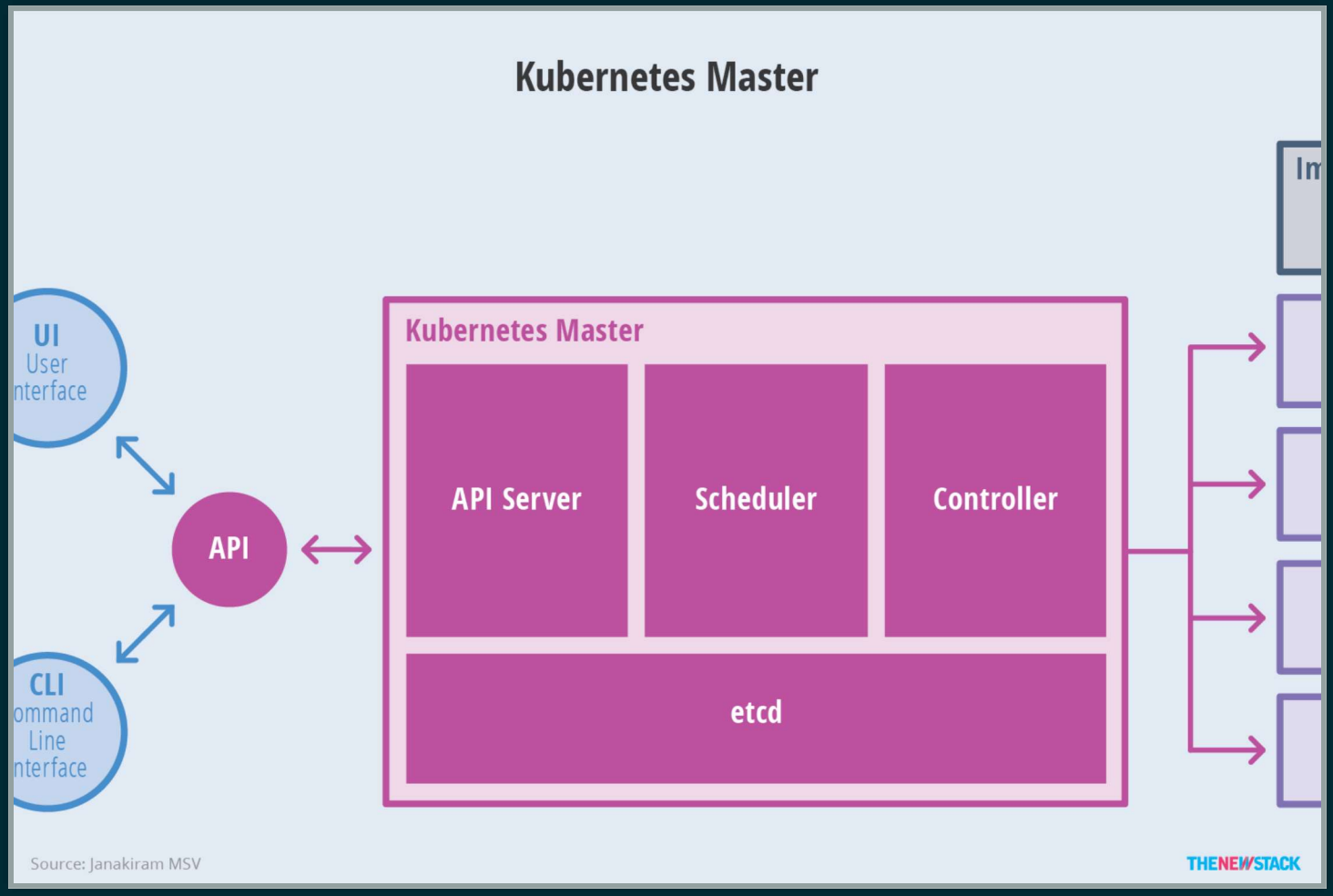
...services, support, and tools are widely available.

KUBERNETES ARCHITECTURE



- Master: Manages the cluster and exposes an API for control.
- Worker: Runs workloads and all of the supporting components.

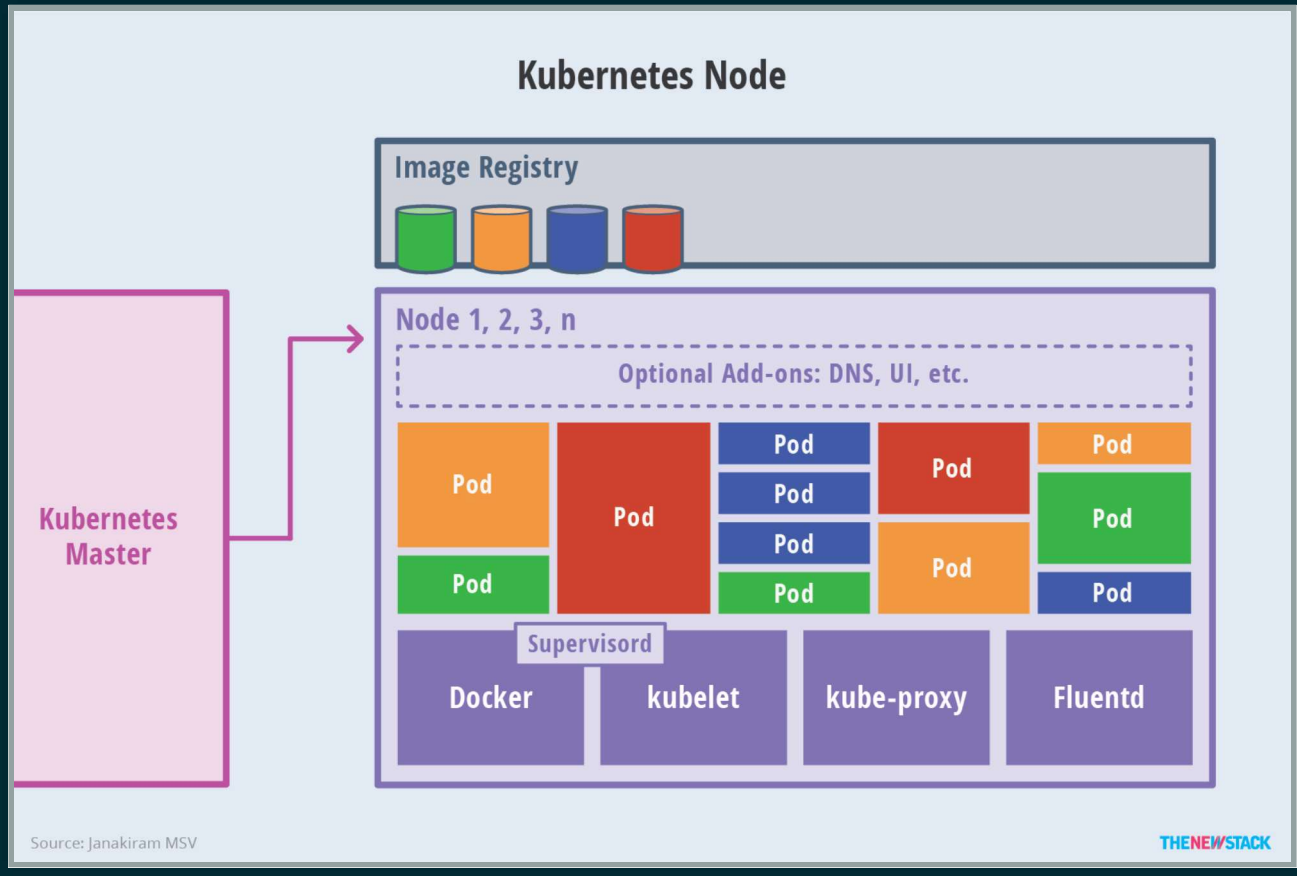
THE INNER WORKINGS OF THE MASTER



- API Server
- Scheduler
- Controller
- Datastore

THE INNER WORKINGS OF A NODE

- kubelet
- kube-proxy
- Container runtime
- Add-ons



COMBINING THE RESOURCE PLANES

BENEFITS

- Fewer machines means lower cost.

DRAWBACKS

- Potential performance issues with the etcd and master components.
- Fault tolerance of the data plane is limited to one less than half of the total number of machines.

SEPARATING THE RESOURCE PLANES

BENEFITS

- Master machines less likely to experience losses from Worker machine behavior.
- Worker machines can be sized appropriately for workloads.

DRAWBACKS

- Higher cost when running more machines.

BEST PRACTICES

For a production environment, Rancher recommends:

- Isolated compute, data, and worker planes
- Regular backups of `etcd`

KUBERNETES RESOURCES

Kubernetes objects are called *resources*. Today we'll be working with:

- Pods
- Deployments
- Services
- Ingresses

POD

Pods are the smallest unit that can be deployed in Kubernetes.

They consist of one or more containers that are always scheduled together.

Each pod is given a unique IP address.

Containers in a pod communicate with each other via localhost.

A group of identical pods is known as a ReplicaSet.

DEPLOYMENT

Deployments are a level of abstraction above ReplicaSets

They create and update ReplicaSets.

They allow you to easily scale and perform rolling updates to workloads.

SERVICE

A Service defines a DNS entry that points to a ReplicaSet.

It provides a consistent endpoint for the ReplicaSet.

Services exist in different forms:

- NodePort
- ClusterIP
- LoadBalancer

INGRESS

An Ingress defines how traffic from outside of the cluster is routed to resources within the cluster.

It exposes one or more Services to the world.

It usually handles HTTP traffic and routes it internally based on factors such as host and path.

It is usually implemented by a software load balancer (Nginx, HAProxy, Traefik, etc.)

INTERMISSION

MACHINE PROVISIONING

VAGRANT

Clone the repository and deploy the virtual machines.

```
git clone https://github.com/rancher/rodeo && cd rodeo/vagrant
vagrant up
```

The Rancher Server will be available at https://172.22.101.101

DIGITAL OCEAN

Clone the repository and rename the `tfvars` file.

```
git clone https://github.com/rancher/rodeo && cd rodeo/do
mv terraform.tfvars.example terraform.tfvars
```

Edit `terraform.tfvars` and then deploy the instances.

```
terraform init
terraform apply
```

This will:

- Start a droplet for the server
- Start `{count_agent_all_nodes}` amount of droplets for Kubernetes

AWS

Clone the repository and rename the `tfvars` file.

```
git clone https://github.com/rancher/rodeo && cd rodeo/aws
mv terraform.tfvars.example terraform.tfvars
```

Edit `terraform.tfvars` and then deploy the instances.

```
terraform init
terraform apply
```

This will:

- Start an EC2 instance for the server
- Start `{count_agent_all_nodes}` amount of EC2 instances for Kubernetes

ROLL YOUR OWN

Deploy one Ubuntu 16.04 node for the Rancher Server

Deploy one to three Ubuntu 16.04 nodes for the Kubernetes cluster

Install Docker 17.03.x:

```
curl https://releases.rancher.com/install-docker/17.03.sh | sh
```

REQUIRED PORTS

For today, make sure that the following ports are open:

TO THE RANCHER SERVER

- ANY -> 80/443/tcp - Rancher UI
- ANY -> 22/tcp - SSH
- Nodes -> 2049/tcp - NFS

TO THE KUBERNETES NODES

- ANY -> 80/443/tcp - Ingress
- ANY -> 22/tcp - SSH
- Other Nodes -> ANY
- Rancher Server -> ANY

DEPLOYING RANCHER SERVER

INSTALLATION TYPES

- Single Node / Standalone
- High Availability

SERVER TAGS

- `latest` - The latest release. Not suitable for production.
- `stable` - Use this in production. Unexpected bugs from `latest` have been fixed.

DEPLOYING RANCHER SERVER

Rancher Server runs as a Docker container:

```
ssh {server_ip}
docker run -d --restart=unless-stopped \
    -p 80:80 -p 443:443 \
    -v /opt/rancher:/var/lib/rancher \
    rancher/rancher:stable
```

Open a browser and point it at *https://{server_ip}/*.

INTRODUCING THE RANCHER UI

GLOBAL CONFIGURATION

- Add clusters
- Configure additional node drivers
- Add external application catalogs
- Configure authentication and security

SECURITY

- Pull users and groups from approved backend authentication providers
- Collect namespaces into projects
- Assign group policy to projects
- Extend group policy uniformly across all clusters
- Utilize Pod Security Policies for additional control

WORKLOADS

- Workloads (Deployment, DaemonSet, StatefulSet, Cron, Job)
- Load Balancing
- Service Discovery
- Volumes
- Pipelines

CATALOG APPLICATIONS

- Rancher-curated or private Helm repositories for application deployment
- Provides a way to package applications that others can deploy

RESOURCES

- Certificates
- ConfigMaps and Secrets
- External Registries
- Logging
- Alerts

COFFEE BREAK

DEPLOYING KUBERNETES

ADDING A CLUSTER

- From the Global screen, select *Add Cluster*
- Choose *Custom*
- Give the cluster a name and click *Next*
- Select all three roles and copy the Docker command
  - **Note:** If your nodes are in a cloud provider and are using private IPs, or if you're using Vagrant, expand *Show advanced options* and set the public and private addresses for the node.
- SSH into your Kubernetes nodes and execute the Docker command.
- If using multiple nodes, repeat this process, changing roles or address information as necessary.

INTERACTING WITH THE CLUSTER

USING RANCHER TO DEPLOY WORKLOADS

- Select the *Default* project for the cluster
- Select *Deploy*
- Fill in the appropriate fields
  - Set the name to `rodeo`
  - Set the Docker Image to `nginx:alpine`
  - Add Port 80 as type `ClusterIP`
- Click *Launch*

WHAT RESOURCES DID WE CREATE?

- A Deployment
- A ReplicaSet
- A Service

DEPLOY AN INGRESS

- Select the *Load Balancing* tab and then select *Add Ingress*
- Give it the name *rodeo*.
- Delete the rule that Rancher made for you and add a *Service* rule.
- Route / to the rodeo Service on port 80 and click *Save*.
- Wait for the Ingress to be created, and click its link.

INTRODUCTION TO KUBECTL

From the cluster overview page, click *Launch kubectl*

Some commands that you can try:

```
kubectl get nodes
kubectl get deployment
kubectl get pods
kubectl describe po/{pod_name}
```

USING KUBECTL TO UPGRADE A WORKLOAD

```
kubectl set image deploy/rodeo rodeo=apache:alpine
kubectl get pods
kubectl rollout status deploy/rodeo
```

USING RANCHER TO UPGRADE A WORKLOAD

- Select *Workloads* and then select *Edit* from the menu on the right side of your workload.
- Update the image to `superseb/rancher-demo` and change the container port to `8080`
- Click *Upgrade*
- Update the Ingress to use port `8080` on the Service
  - The UI will automatically pick up the change in the service and update the field to 8080 for you. Just click *Save*.

SCALING WORKLOADS

USING RANCHER

- From the *Workloads* page, click the arrow on the left side of your workload to expand it.
- Click the *+* to scale your workload.

USING KUBECTL

```
kubectl scale deploy/rodeo --replicas=3
```

REVIEW

When we create a Workload, what gets created in Kubernetes?

> *A Deployment, a ReplicaSet, and, if requested, a Service and storage resources.*

How can we update the running image in a pod's container?

> *By editing the workload config in the Rancher UI, or by using `kubectl set image`*

APPLICATION DEPLOYMENT

DEPLOYING WORDPRESS FROM THE CATALOG

- Select *Launch* from the *Catalog Apps* tab. Scroll down to *Wordpress* and select *View Details*.
- Set the following additional parameters:
  - Username: admin
  - Password: {something you'll remember}
  - Admin Email: {your email}
- Click *Launch*.

WHAT'S HAPPENING IN THE BACKGROUND?

Rancher created the `wordpress` namespace

Into this namespace it deployed several configmaps, secrets, `mariadb` and `wordpress`

It created an Ingress listening on an `xip.io` domain.

CREATE SOME CONTENT!

Follow the `xip.io` link to your new blog. Scroll down to the *Log In* link, log in, and post some content.

INTERMISSION

- What questions do you have about what we've learned so far?
- Do we have enough time to cover the advanced rodeo topics?
- If not, we can wrap it up.

ADVANCED TOPICS

WHAT NEXT?

You have a cluster. It's running an application. What do you want to do next?

- Persistent Storage
- Monitoring with Prometheus and Grafana
- Using the Rancher CLI

What would you like to learn?

PERSISTENT STORAGE

WHY DO WE NEED THIS?

- Containers are ephemeral.
- Design for failure.
- Some applications benefit from shared storage.
- Countless other reasons.

LET'S FAIL SOMETHING!

Run the following to simulate a failure of the database:

```
kubectl -n wordpress delete po/wordpress-mariadb-0
```

Watch as Kubernetes terminates it and recreates it and the Wordpress pod again.

```
kubectl -n wordpress get po
```

When the pods have been recreated, reload your blog post. What's different?

INTRODUCING STORAGE CLASSES

A storage class provides a way for administrators to describe the "classes" of storage they offer. Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators.

- From the Kubernetes documentation

BUILDING AN NFS STORAGECLASS PROVISIONER

PART ONE: DEPLOYING THE SERVER

On the Rancher Server, make a directory to hold persistent data, and then share that directory via an NFS server container:

```
sudo apt -qq -y install nfs-kernel-server

sudo systemctl stop nfs-kernel-server

sudo systemctl disable nfs-kernel-server

sudo mkdir -p /opt/nfs

sudo docker run -d --name nfs --restart=always \
  --privileged --net=host -v /opt/nfs:/exports -e \
  SHARED_DIRECTORY=/exports itsthenetwork/nfs-server-alpine:4
```

If you haven't already opened port `2049/tcp` on the server, do so now in your security policy configuration.

BUILDING AN NFS STORAGECLASS PROVISIONER

PART TWO: DEPLOYING THE PROVISIONER

In the Rancher UI, Select the System project and create a new Namespace called `nfs-client-provisioner`.

Next, import the `provisioner.yaml` file from the `k8s` folder in the `rancher/rodeo` github repository. Change the NFS server address in both places to match your NFS server location.

Verify the installation was successful:

```
> kubectl -n nfs-client-provisioner get po
NAME                                     READY     STATUS    RESTARTS   AGE
nfs-client-provisioner-865b7bc646-n7zjs   1/1       Running   0          2m
```

USING THE STORAGECLASS FOR PERSISTENT STORAGE

Delete and redeploy Wordpress from the App Catalog, setting both MariaDB and Wordpress to create persistent volumes. Tell it to use the `wordpress` namespace, or else it will create a new one.

You can look in `/opt/nfs` on the Rancher Server node to see the volumes as they're created.

CAN WE SURVIVE A FAILURE?

- Create a post in your new Wordpress installation.
- Delete the database container and let it redeploy.
- Did your content survive?

MONITORING WITH PROMETHEUS AND GRAFANA

WHAT IS PROMETHEUS?

Prometheus is a monitoring solution for systems and services.

It uses a pull model to fetch data at specific intervals.

It then processes that data, evaluates expressions, takes action, sends alerts, or does other things that it might need to do.

WHAT IS GRAFANA?

Grafana is a graphing and dashboard tool.

It can communicate with multiple backends, such as

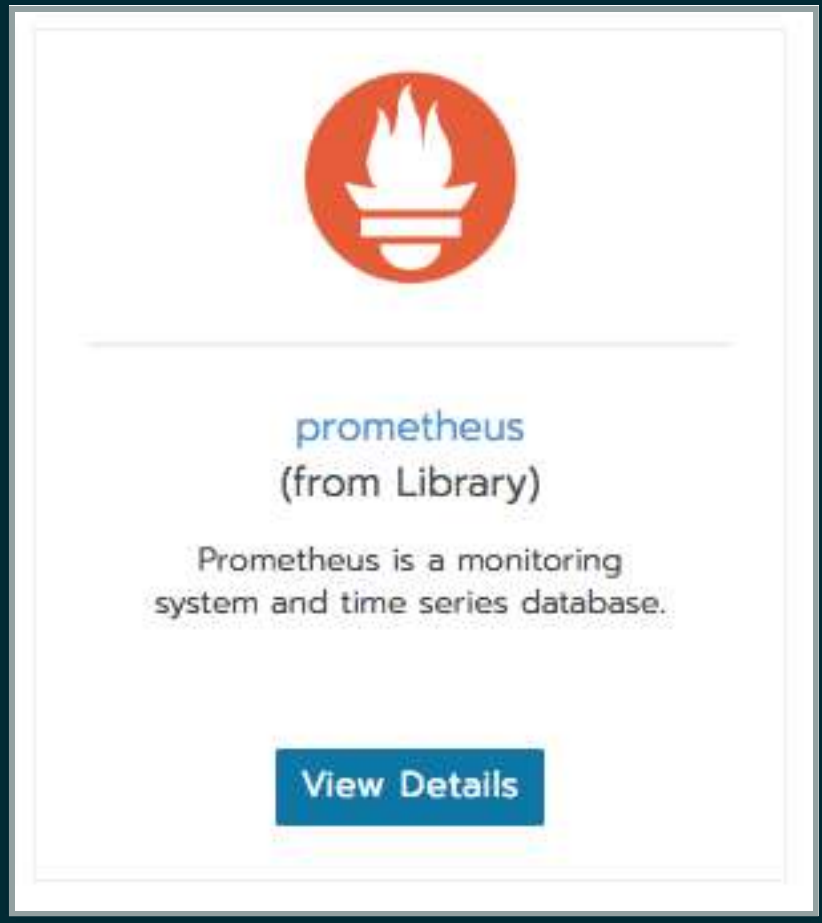
- Elasticsearch
- Graphite
- OpenTSDB
- Databases
- Prometheus

Grafana makes beautiful graphs and dashboards.

GRAFANA DASHBOARDS ARE POWERFUL

INSTALLING PROMETHEUS AND GRAFANA

- Install the *prometheus* app from the App Catalog.
- Leave the settings at their defaults.
- When the deployment is complete, click on the *prometheus-grafana* target under *Load Balancing*

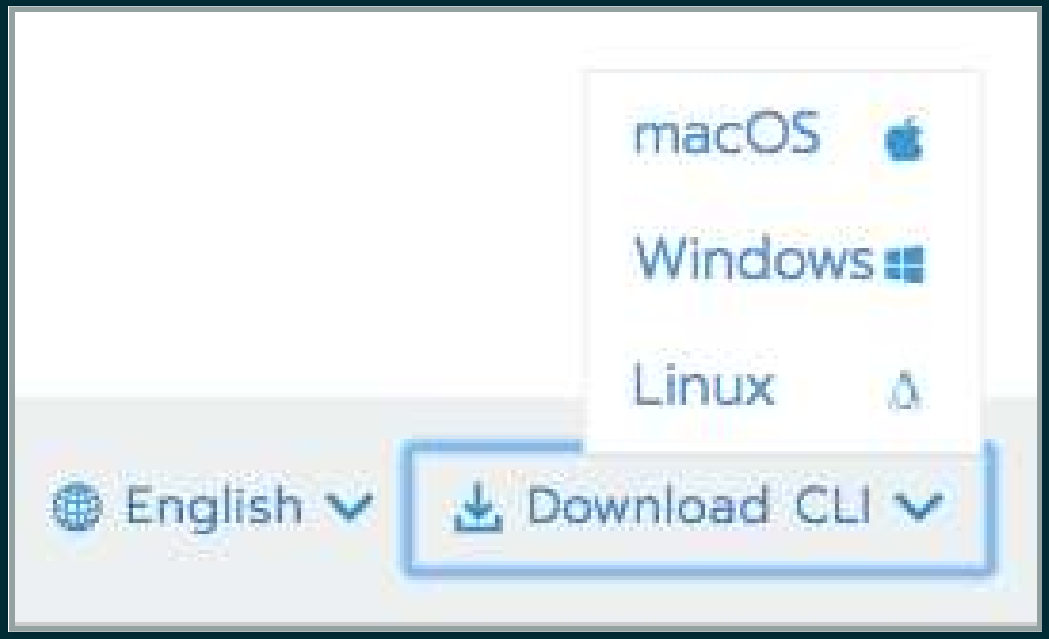


USING THE RANCHER CLI

WHY CLI?

- The non-kubectl functionality of the GUI is available via the CLI.
- Some tasks are easier via the CLI.
- The CLI integrates with external applications that can't use the GUI, like CI/CD systems, monitoring systems, etc.

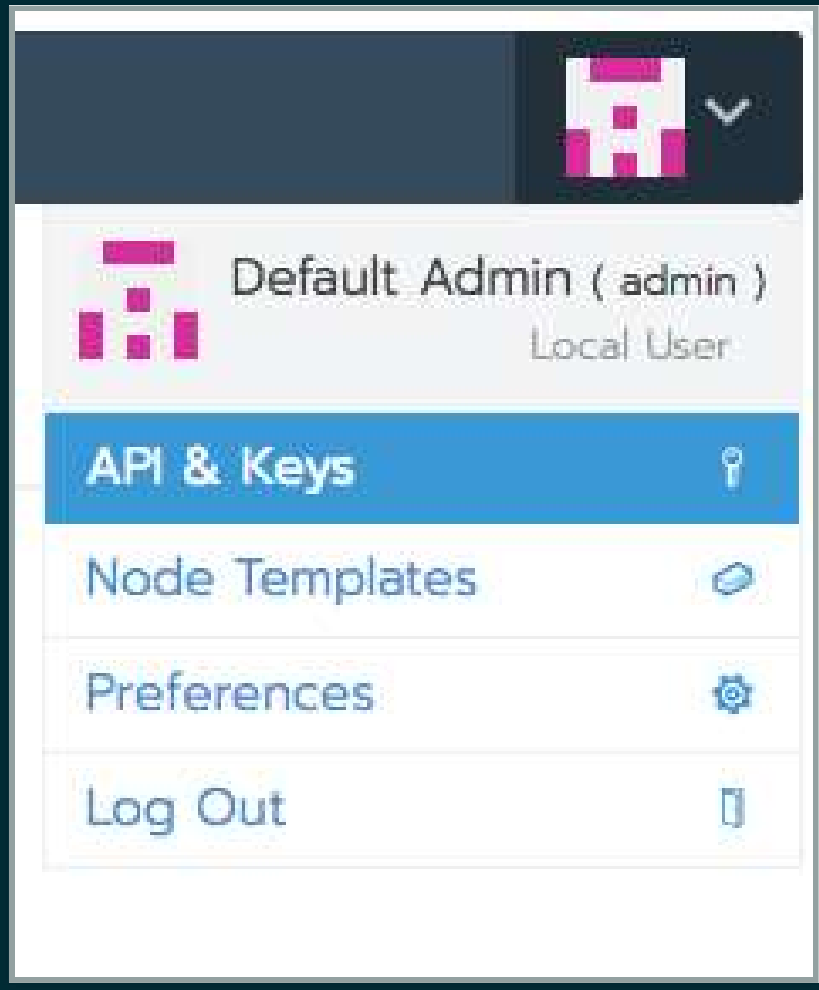
DOWNLOADING THE CLI



From your Rancher Server GUI, follow the link in the bottom right corner to download the CLI binary for your platform. Extract the contents and move it into your path.

CREATING AN API KEY

- Click on the avatar image in the top right and select *API & Keys* from the dropdown menu that appears.
- Click *Add Key*
- Enter a name for your key (such as your username) and click *Create*



Ordinarily you would securely store this information. For today, you can copy the Bearer Token to your clipboard.

AUTHENTICATING WITH THE CLI

Use the *Bearer Token* from the last step to authenticate with your Rancher server:

```
$ ./rancher login https://{SERVER_URL} --token {BEARER_TOKEN}
```

Rancher will ask you to accept the self-signed certificate and then will ask you to choose a project. Choose the *Default* project.

You can change projects in the future with `rancher context switch`.

USING THE CLI

- `rancher kubectl ...`
- `rancher inspect ...`
- `rancher project create myproject`
- `rancher ps`

CLOSING