HashiCorp

Vagrant

Development Environments Made Easy

Allan MacLean

allan@yellowzone.co.uk

HashiCorp Interview - February 2020

https://github.com/yellowzoneallan/HashiCorp

X86 Virtualisation - 1991

Acorn

- Acorn Archimedes A5000
- 25MHz ARM3 processor
- 4Mb RAM, 80Mb harddrive
- Emulated 8086 - PC Dos
- SPICE3 & MATLAB
- Risc PC 486 Card

X86 Virtualisation - 2020

- Apple MacBook Pro 16”
- 2.3GHz Intel i9 8-core CPU
- 64Gb RAM, 1TB harddrive
- Vagrant
- VirtualBox

Infrastructure as Code Tools

Configuration Management

HashiCorp Vagrant

Hypervisor

VirtualBox

IDE / Editor

Visual Studio Code

vi

Target OS

ubuntu

CentOS

app.vagrantup.com

HashiCorp Vagrant Cloud

Search | Pricing | Vagrant | Help | Create an Account | Sign In

Discover Vagrant Boxes

wordpress

Provider: any | virtualbox | vmware | libvirt | more

Sort by: Downloads | Recently Created | Recently Updated

dsadovnichy/lemp_wp_centos7 0.0.1 centOS 7, nginx, mariaDB, php 7.3, wordpress 5.3.2	virtualbox	Downloads 0	Released 2 days ago
seravo/wordpress 20200130.0.0 Seravo WordPress Stack for local development. For details see https://seravo.com/docs/	libvirt, virtualbox	Downloads 22,235	Released 9 days ago
dsadovnichy/lemp_wp 0.0.1 ubuntu 18.04, nginx, mysql 5.7, php 7.2, wordpress 5.3.2	virtualbox	Downloads 2	Released 10 days ago

Vagrant Commands

Vagrantfile

- vagrant up
- vagrant ssh
- vagrant ssh <instance name>
- vagrant destroy -f

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/disco64"
  config.vm.provider "virtualbox" do |v|
    v.memory = 2048
    v.cpus = 2
  end
  config.vm.network :private_network, ip: "192.168.3.3"
  config.vm.provision "shell", inline: <<-SHELL
     apt-get update -y
     apt-get install -y nginx
     systemctl enable nginx
     systemctl start nginx
   SHELL
end
```

Can Vagrant help me play with tech stacks?

- Quickly, cleanly, examine and evaluate products in a repeatable manner
- No residue configurations, no network funny's, or disk space retained on my laptop after use

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "centos/7"
  config.vm.provider "virtualbox" do |v|
    v.memory = 2048
    v.cpus = 2
  end
  config.vm.network :private_network, ip: "192.168.3.3"
  config.vm.provision "shell", inline: <<-SHELL
     yum update -y
     yum install -y httpd
     systemctl start httpd
     systemctl enable httpd
   SHELL
end
```

VM remote HTTP access

Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/disco64"
  config.vm.provider "virtualbox" do |v|
    v.memory = 2048
    v.cpus = 2
  end
  config.vm.network :private_network, ip: "192.168.3.3"
  config.vm.provision "shell", inline: <<-SHELL
     apt-get update -y
     apt-get install -y nginx
     systemctl enable nginx
     systemctl start nginx
   SHELL
end
```

VM remote HTTP access

Can Vagrant help me isolate multiple projects or customers?

- Build multiple identical environments
- 100% isolation of AWS/Azure/GCP keys & secrets
- 0% chance of contamination between environments

command line versions

```
vagrant@ubuntu-eoan:~$ python3 --version
Python 3.7.5
vagrant@ubuntu-eoan:~$ ansible --version | head -1
ansible 2.8.3
vagrant@ubuntu-eoan:~$ sudo docker --version
Docker version 19.03.2, build 6a30dfca03
vagrant@ubuntu-eoan:~$ aws --version
aws-cli/1.16.218 Python/3.7.5 Linux/5.3.0-29-generic botocore/1.12.208
vagrant@ubuntu-eoan:~$ gcloud --version
Google Cloud SDK 279.0.0
alpha 2020.01.31
beta 2020.01.31
bq 2.0.53
core 2020.01.31
gsutil 4.47
kubectl 2020.01.31
vagrant@ubuntu-eoan:~$ az --version
azure-cli                         2.0.81
command-modules-nspkg              2.0.3
core                              2.0.81
nspkg                              3.0.4
telemetry                          1.0.4
```

Vagrantfile

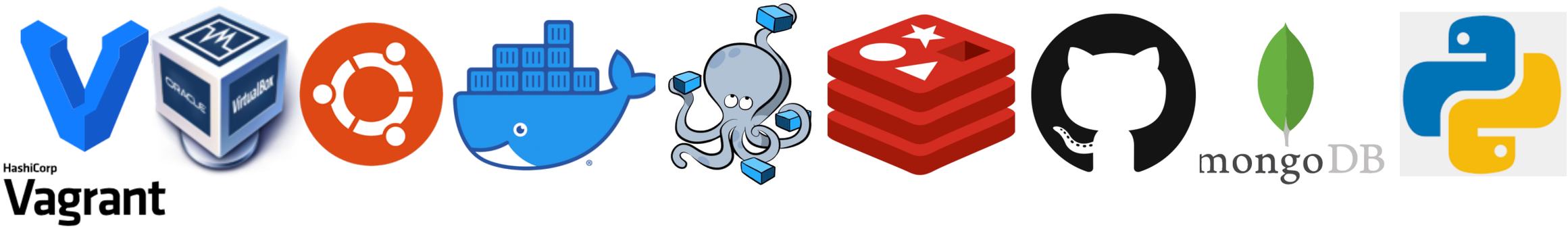
```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/eoan64"
  config.vm.provider "virtualbox" do |v|
    v.memory = 2048
    v.cpus = 2
  end
  config.vm.network :private_network, ip: "192.168.3.3"
  config.vm.provision "shell", path: "setup.sh"
end
```

setup.sh

```
sudo apt-get update -y # patch linux
sudo apt-get install -y curl; sudo apt-get install -y unzip # install unix tools
sudo apt-get install -y python3-pip # install python
sudo apt install -y docker.io; sudo systemctl start docker; sudo systemctl enable docker
sudo apt install -y net-tools # install unix tools
sudo apt install -y wget # install unix tools
sudo snap install terraform # install terraform (not latest)
sudo apt install -y awscli # install aws tools
sudo snap install google-cloud-sdk --classic # install gcp tools
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash # install azure tools
# fetch terraform provider stuff for each cloud
git clone https://github.com/terraform-providers/terraform-provider-aws.git
git clone https://github.com/terraform-providers/terraform-provider-azurerm.git
git clone https://github.com/terraform-providers/terraform-provider-google.git
git clone https://github.com/terraform-providers/terraform-provider-kubernetes.git
sudo apt install -y ansible # install ansible
```

Can Vagrant provision me a development environment?

- Build multiple isolated development environments
- Setup a development environment that runs great even without internet connectivity

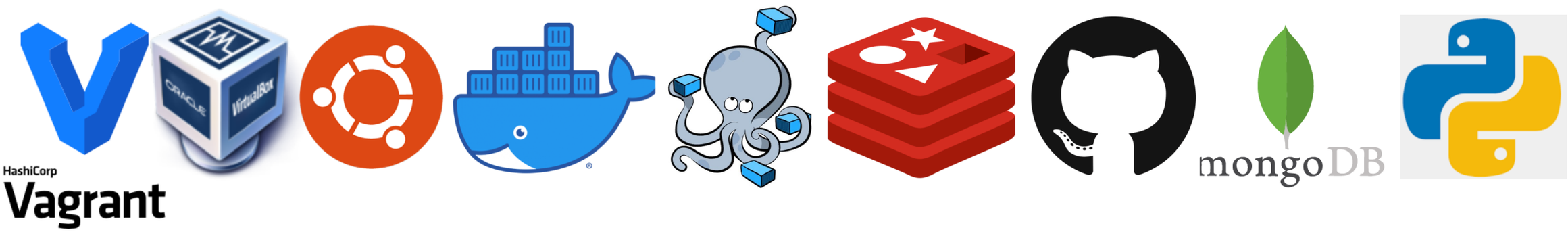


Vagrantfile

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/disco64"
  config.vm.provider "virtualbox" do |v|
    v.memory = 4096
    v.cpus = 2
  end
  config.vm.network :private_network, ip: "192.168.3.3"
  config.vm.provision "shell", inline: <<-SHELL
  git clone https://github.com/yellowzoneallan/MAT; cd MAT; ./setup.sh
   SHELL
end
```

sudo docker stats

```
vagrant@ubuntu-disco:~$ sudo docker stats
CONTAINER ID        NAME                CPU %.  MEM USAGE / LIMIT    MEM %   NET I/O             BLOCK I/O           PIDS
09fc3473ef08        mat-python-v1       0.76%   16.93MiB / 3.85GiB   0.43%   1.92MB / 3.05MB     1.44MB / 0B         5
899f6f18d238        webapp              0.03%   5.414MiB / 3.85GiB   0.14%   1.38kB / 0B         0B / 0B             6
f499ee57f44f        source_gps          1.22%   137.6MiB / 3.85GiB   3.49%   109kB / 399kB       0B / 32.8kB         25
3d4e6faa73a9        mqtt-to-websocket   1.96%   343.1MiB / 3.85GiB   8.70%   1.05MB / 403kB      0B / 0B             46
6b77cad27cc4        redis               0.19%   2.551MiB / 3.85GiB   0.06%   1.84MB / 1.3MB      0B / 1.29MB         4
5341b23481bd        broker              0.96%   9.445MiB / 3.85GiB   0.24%   1.26MB / 1.53MB     24.6kB / 0B         1
b5f785897945        mongodb             0.27%   69.87MiB / 3.85GiB   1.77%   753kB / 244kB       2.22MB / 2.88MB     34
```



setup.sh

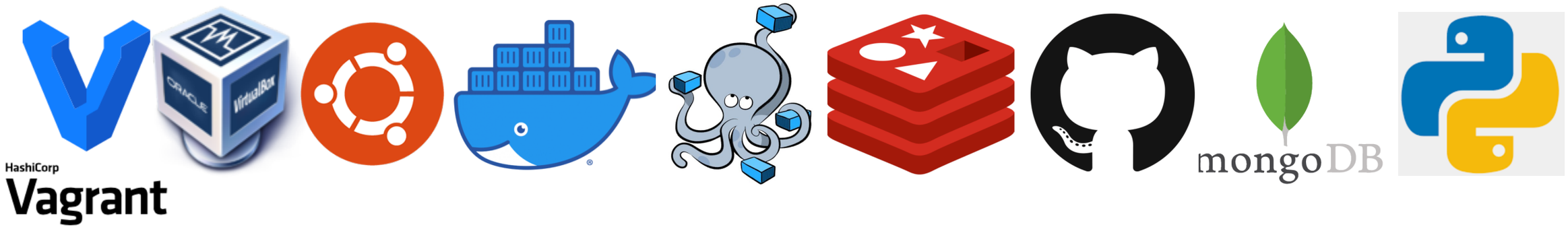
```
sudo apt-get update -y; # patch linux
sudo apt-get install -y curl; sudo apt-get install -y unzip # install unix tools
sudo apt-get install -y mosquitto; sudo apt-get install -y mosquitto-clients # install MQTT tools
sudo apt-get install -y python3-pip;  sudo pip3 install paho-mqtt; sudo pip3 install geopy # install python tools
sudo apt install  -y docker.io; sudo systemctl start docker; sudo systemctl enable docker # install and start docker
sudo curl -L "https://github.com/docker/compose/releases/download/1.25.0/docker-compose-$(uname -s)-$(uname -m)"
-o /usr/local/bin/docker-compose; sudo chmod +x /usr/local/bin/docker-compose # install docker compose
sudo service  mosquitto stop # stop MQTT - port conflict
sudo pip3 install redis # install redis cache tools
sudo apt install -y net-tools # install unix netwoking tools
sudo pip3 install pymongo; sudo apt install -y mongodb-clients # install mongodb tools
sudo docker build --tag mat-python-v1 .
sudo docker-compose down
sudo docker-compose up -d
```

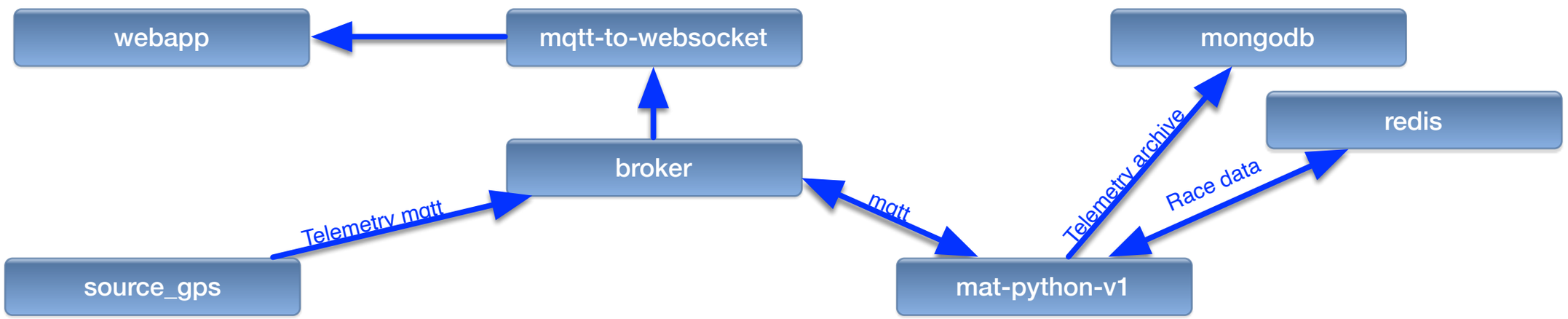
docker-compose.yaml

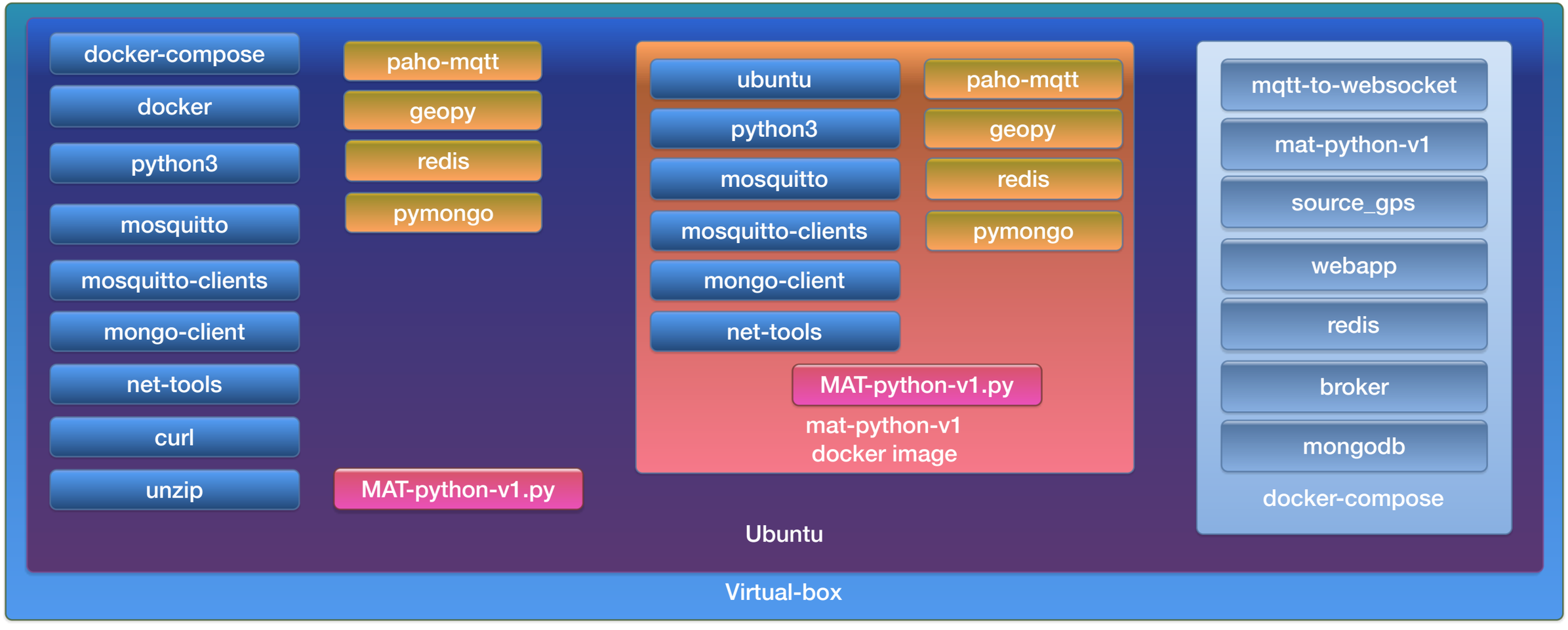
MAT-python-v1.py

Dockerfile

```
FROM ubuntu:latest
RUN apt-get update -y
RUN apt-get install -y mosquitto
RUN apt-get install -y mosquitto-clients
RUN apt-get install -y python3-pip
RUN apt-get install -y mongodb-clients
RUN apt-get install -y curl
RUN apt-get install -y net-tools
RUN pip3 install paho-mqtt
RUN pip3 install geopy
RUN pip3 install redis
RUN pip3 install pymongo
COPY . /app
WORKDIR /app
EXPOSE 5555
CMD /usr/bin/python3 ./MAT-python-v1.py
```


HashiCorp
Vagrant
mongoDB


webapp
mqtt-to-websocket
mongodb
redis
broker
Telemetry archive
Race data
Telemetry mqtt
mqtt
source_gps
mat-python-v1


docker-compose
docker
python3
mosquitto
mosquitto-clients
mongo-client
net-tools
curl
unzip
paho-mqtt
geopy
redis
pymongo
MAT-python-v1.py
ubuntu
python3
mosquitto
mosquitto-clients
mongo-client
net-tools
paho-mqtt
geopy
redis
pymongo
MAT-python-v1.py
mat-python-v1
docker image
mqtt-to-websocket
mat-python-v1
source_gps
webapp
redis
broker
mongodb
docker-compose
Ubuntu
Virtual-box

Show me what you do with Vagrant?

- 20 years automating infrastructure and deployments, nowadays referred to as “Infrastructure as Code”, first attempt is always in Vagrant
- Vagrant allows me to have my first few iterations in private, nobody sees the big failures. All the mistakes are made without audit logs, outages, or cloudy bills!

Internet Banking

Vagrantfile

bootstrap.sh

bootstrap_was_dm.sh

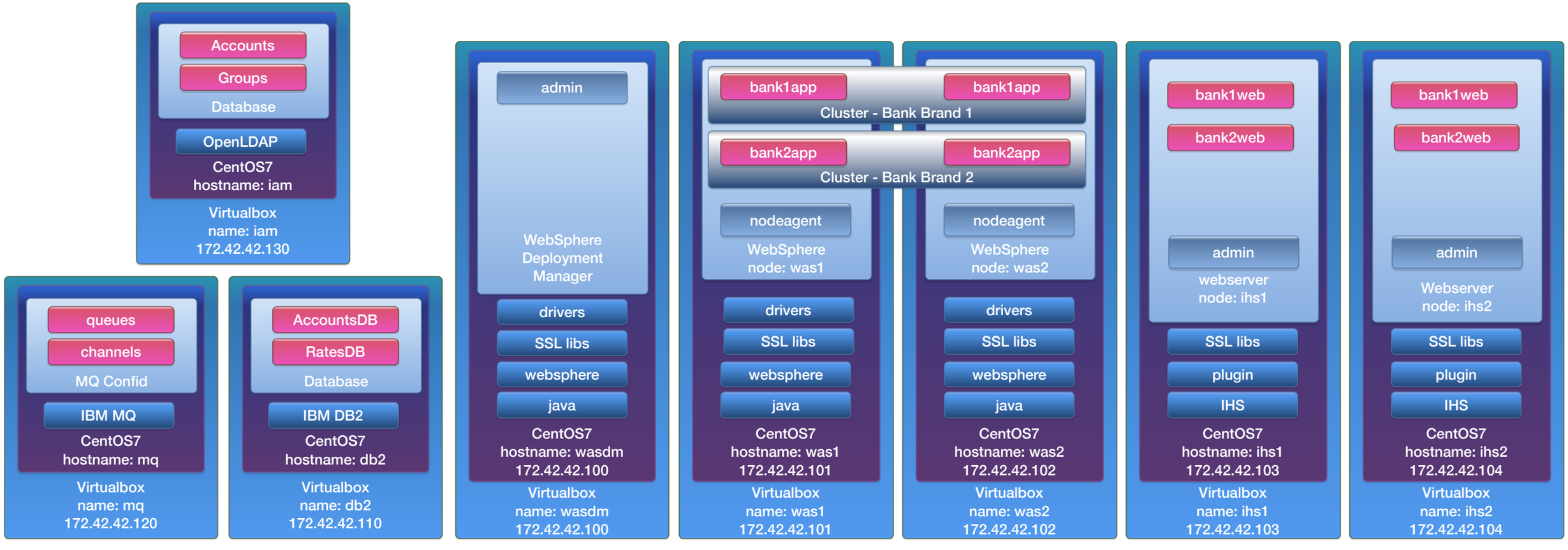
bootstrap_was_node.sh

bootstrap_ihs_node.sh

bootstrap_db2.sh

bootstrap_mq.sh

bootstrap_ldap.sh



Isn’t the future docker & microservices?

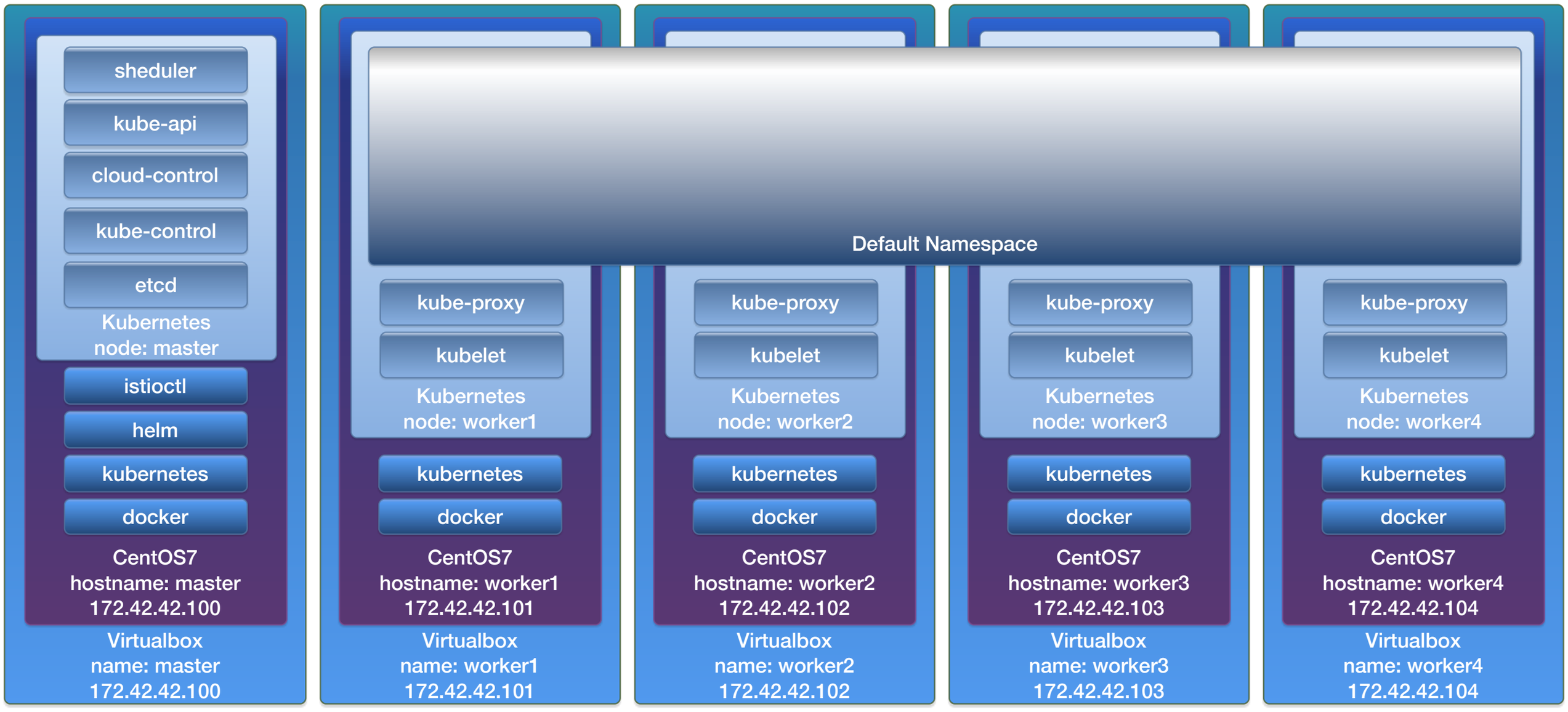
Kubernetes

Vagrantfile

bootstrap_master.sh

bootstrap.sh

bootstrap_worker.sh



Vagrant Kubernetes Build

Vagrantfile

```
# vi: set ft=ruby :
ENV['VAGRANT_NO_PARALLEL'] = 'yes'
Vagrant.configure(2) do |config|
  config.vm.provision "shell", path: "bootstrap.sh"
  # Kubernetes Master Server
  config.vm.define "master" do |master|
    master.vm.box = "centos/7"
    master.vm.hostname = "master.example.com"
    master.vm.network "private_network", ip: "172.42.42.100"
    master.vm.provider "virtualbox" do |v|
      v.name = "master"
      v.memory = 2048
      v.cpus = 2
    end
    master.vm.provision "shell", path: "bootstrap_master.sh"
  end
  NodeCount = 4
  # Kubernetes Worker Nodes
  (1..NodeCount).each do |i|
    config.vm.define "worker#{i}" do |workernode|
      workernode.vm.box = "centos/7"
      workernode.vm.hostname = "worker#{i}.example.com"
      workernode.vm.network "private_network", ip: "172.42.42.10#{i}"
      workernode.vm.provider "virtualbox" do |v|
        v.name = "worker#{i}"
        v.memory = 2048
        v.cpus = 1
      end
      workernode.vm.provision "shell", path: "bootstrap_worker.sh"
    end
  end
end
```

bootstrap.sh

```
Allans-MBP:hashicorp allanmaclean$ cat bootstrap.sh | grep TASK
echo "[TASK 1] Update /etc/hosts file"
echo "[TASK 2] Install docker container engine"
echo "[TASK 3] Enable and start docker service"
echo "[TASK 4] Disable SELinux"
echo "[TASK 5] Stop and Disable firewalld"
echo "[TASK 6] Add sysctl settings"
echo "[TASK 7] Disable and turn off SWAP"
echo "[TASK 8] Add yum repo file for kubernetes"
echo "[TASK 9] Install Kubernetes (kubeadm, kubelet and kubectl)"
echo "[TASK 10] Enable and start kubelet service"
echo "[TASK 11] Enable ssh password authentication"
echo "[TASK 12] Set root password"
```

bootstrap_master.sh

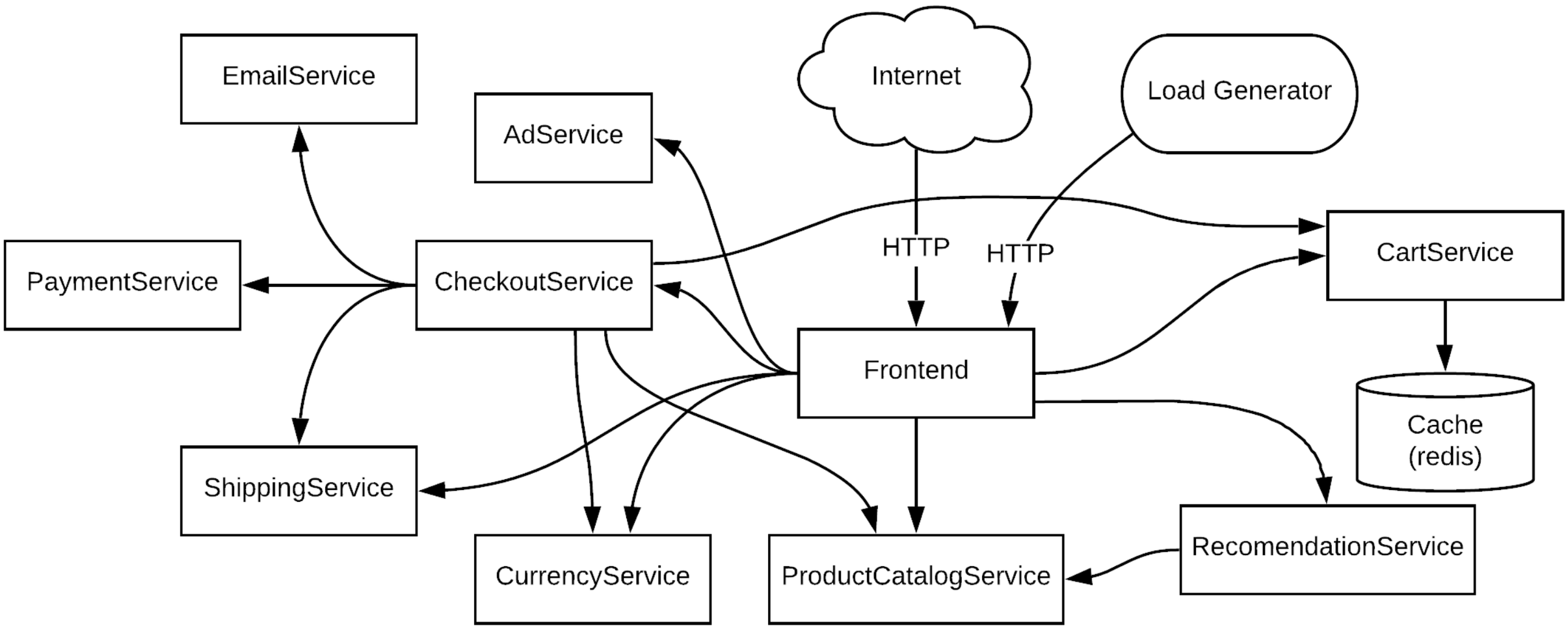
```
Allans-MBP:hashicorp allanmaclean$ cat bootstrap_master.sh | grep TASK
echo "[TASK 1] Initialize Kubernetes Cluster"
echo "[TASK 2] Copy kube admin config to Vagrant user .kube directory"
echo "[TASK 3] Deploy Calico network"
echo "[TASK 4] Generate and save cluster join command to /joincluster.sh"
echo "[TASK 5] Install helm 3"
echo "[TASK 6] Install istio binary"
```

bootstrap_worker.sh

```
Allans-MBP:hashicorp allanmaclean$ cat bootstrap_worker.sh | grep TASK
echo "[TASK 1] Join node to Kubernetes Cluster"
```

Microservices Architecture

https://github.com/GoogleCloudPlatform/microservices-demo



vagrant ssh master

```
sudo yum install -y git
git clone https://github.com/GoogleCloudPlatform/microservices-demo.git
cd microservices-demo/
vi ./release/kubernetes-manifests.yaml
```

kubernetes-manifests.yaml

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-external
spec:
  type: LoadBalancer
  selector:
    app: frontend
  ports:
  - name: http
    port: 80
    targetPort: 8080
---
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-external
spec:
  type: NodePort
  selector:
    app: frontend
  ports:
  - name: http
    port: 8080
    nodePort: 30080
---
```

Change 3 lines to decouple the GCP Loadbalancer

```
kubectl apply -f ./release/kubernetes-manifests.yaml
```

Kubernetes & Apps

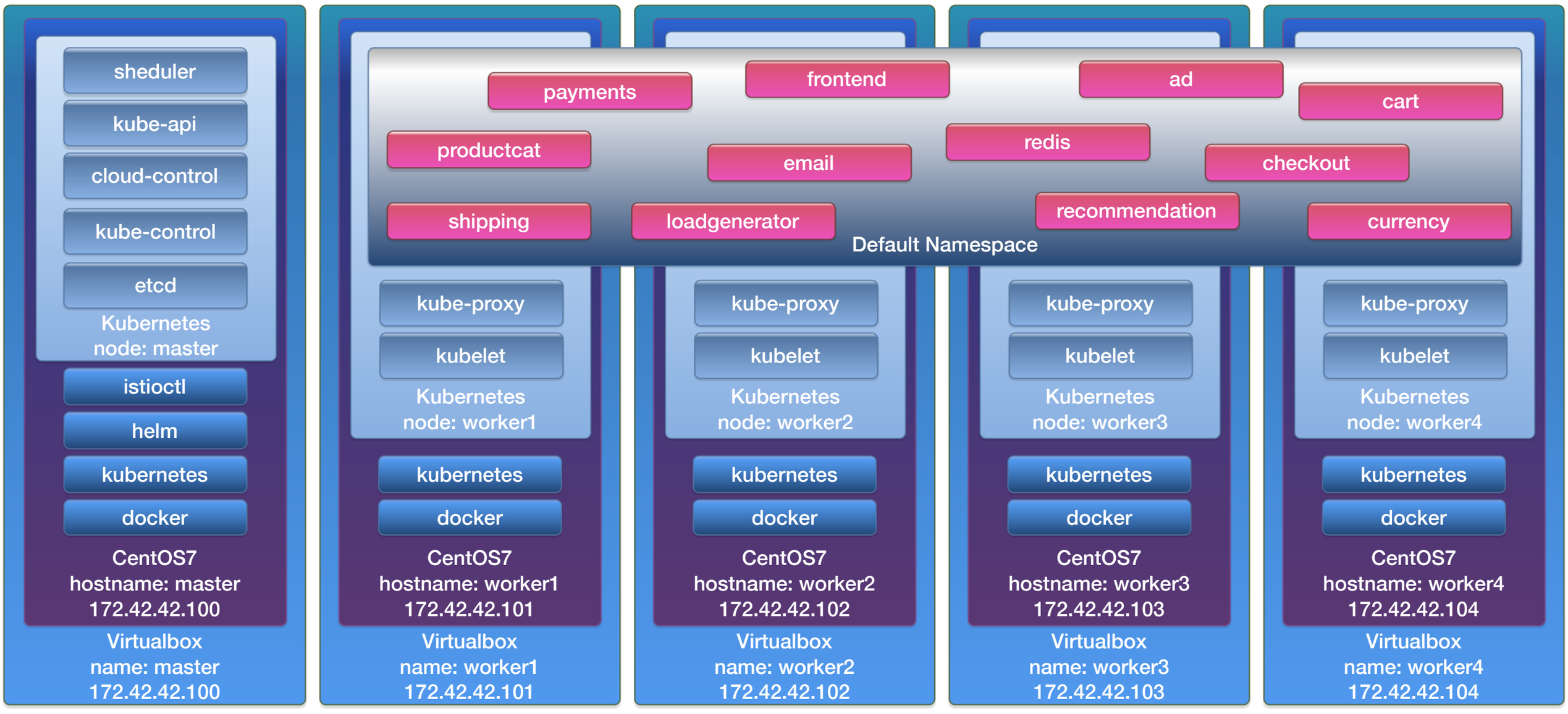
Vagrantfile

bootstrap.sh

bootstrap_master.sh

bootstrap_worker.sh

kubernetes-manifests.yaml



Microservices Example

http://172.42.42.101:30080

Not Secure — 172.42.42.101

Hipster Shop

USD

View Cart (0)

One-stop for Hipster Fashion & Style Online

Tired of mainstream fashion ideas, popular trends and societal norms? This line of lifestyle products will help you catch up with the hipster trend and express your personal style. Start shopping hip and vintage items now!

Vintage Typewriter

Buy

USD 67.98

Vintage Camera Lens

Buy

USD 12.48

Home Barista Kit

Buy

USD 123.99

Microservices & Kubernetes

kubectl get pods

```
[vagrant@master microservices-demo]$ kubectl get pods
NAME                                     READY   STATUS             RESTARTS   AGE
adservice-55f9757757-bmb4g               1/1     Running            0          4m40s
cartservice-684bb46b44-z5s8j             1/1     Running            0          4m40s
checkoutservice-6fcc84467f-rv6nt         1/1     Running            0          4m41s
currencyservice-6c7c479d45-n7lks         1/1     Running            0          4m40s
emailservice-8dd9b76cc-j7lq6             1/1     Running            0          4m41s
frontend-7d8cfc75b5-8jlmd                1/1     Running            0          4m40s
loadgenerator-5db67d555-dhr58            0/1     CrashLoopBackOff   4          4m40s
paymentservice-84ffc75c55-8dmdq          1/1     Running            0          4m40s
productcatalogservice-d564bdf4c-c7d7k    1/1     Running            0          4m40s
recommendationservice-76598d5889-rj58q   1/1     Running            0          4m40s
redis-cart-5f59546cdd-7qdbq              1/1     Running            0          4m40s
shippingservice-b6db65f7f-tmg49          1/1     Running            0          4m40s
```

kubectl get nodes

```
[vagrant@master microservices-demo]$ kubectl get nodes
NAME                  STATUS   ROLES    AGE   VERSION
master.example.com    Ready    master   21m   v1.17.2
worker1.example.com   Ready    <none>   18m   v1.17.2
worker2.example.com   Ready    <none>   16m   v1.17.2
worker3.example.com   Ready    <none>   14m   v1.17.2
worker4.example.com   Ready    <none>   11m   v1.17.2
```

kubectl get services

```
[vagrant@master microservices-demo]$ kubectl get services
NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
adservice               ClusterIP   10.101.201.32    <none>        9555/TCP         3m46s
cartservice             ClusterIP   10.107.212.242   <none>        7070/TCP         3m46s
checkoutservice         ClusterIP   10.110.39.11     <none>        5050/TCP         3m47s
currencyservice         ClusterIP   10.105.193.225   <none>        7000/TCP         3m46s
emailservice            ClusterIP   10.105.47.65     <none>        5000/TCP         3m47s
frontend                ClusterIP   10.102.196.244   <none>        80/TCP           3m46s
frontend-external       NodePort    10.111.42.210    <none>        8080:30080/TCP   3m46s
kubernetes              ClusterIP   10.96.0.1        <none>        443/TCP          21m
paymentservice          ClusterIP   10.100.226.81    <none>        50051/TCP        3m46s
productcatalogservice   ClusterIP   10.107.171.50    <none>        3550/TCP         3m46s
recommendationservice   ClusterIP   10.98.160.146    <none>        8080/TCP         3m46s
redis-cart              ClusterIP   10.105.138.145   <none>        6379/TCP         3m46s
shippingservice         ClusterIP   10.111.130.174   <none>        50051/TCP        3m46s
```

kubectl describe

```
[vagrant@master microservices-demo]$ kubectl desc
Name:                     frontend-external
Namespace:                default
Labels:                   <none>
Annotations:              kubectl.kubernetes.io/l
                            {"apiVersion":"v1","k
external","namespace":"default"},"spec":{"ports":
Selector:                 app=frontend
Type:                     NodePort
IP:                       10.111.42.210
Port:                     http  8080/TCP
TargetPort:               8080/TCP
NodePort:                 http  30080/TCP
Endpoints:                192.168.43.1:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```