Manage your bare-metal infrastructure with a CI/CD-driven approach

Johannes M. Scheuermann
inovex
Johannes M. Scheuermann
Cloud Platform Engineer @ inovex

› Software-Defined Datacenters
› Infrastructure Services
› High Availability / Scalability / Resilience
› CKA / Linux Foundation Trainer

WE’RE HIRING!
inovex.de/jobs
What about you?
Agenda

- Immutable infrastructure
- Kubernetes on bare-metal
- Big Picture
- CI/CD for the infrastructure
- Example Gitlab (CI)
Exercises

Can be found here:

- Github Link: [https://github.com/inovex/fluffy-unicorn](https://github.com/inovex/fluffy-unicorn)
What I assume

- You know what Kubernetes is
- You know the basic components of Kubernetes
- You already did some bare-metal installation
- ..?
What is immutable infrastructure?
“Traditional” infrastructure

- Continuous updates/modification
- Machines get changed after they are created
- Often configuration management is used
- Typically ssh access is required
  - LDAP/Kerberos ...
- Static monitoring
An immutable infrastructure is another infrastructure paradigm in which servers are never modified after they're deployed. If something needs to be updated, fixed, or modified in any way, new servers built from a common image with the appropriate changes are provisioned to replace the old ones. After they're validated, they're put into use and the old ones are decommissioned.

https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure
Immutable infrastructure

- “The” “Golden” image is built
- Never modified after they are deployed
- Update means a new deployment
  - Before you should update the image :)
- No need for ssh access
- Dynamic monitoring
  - Service Discovery needed
Building the “golden” image

- Probably most famous: https://www.packer.io
- Create provider specific images
- Multiple provisioners
- Clean up after provisioning
- Use the distro-specific tooling
- OS image as Code
  - Packer configuration
  - OS configuration
Challenges

- **Image must be kept up-to-date**
  - CVE’s
  - CI/CD for the image

- **New images must be constantly deployed**

- **External configuration is needed in most cases**
  - Certificates
  - Configuration for different stages
  - dev-prod-parity
What we ended up

- Using vanilla Container Linux as base
- Ignition as provisioning tool
  - Only executed once at boot time
- Consul-template + confd for configuration files
- Static certificates are baked into ignition
  - e.g. for secure access to the etcd CMDB
Recap immutable infrastructure

- Vanilla Container Linux ✅
- ignition is only executed once during boot ✅
- Updates means new deployment ✗
  - Until now we didn’t deployed anything :)
- No need for ssh access ✅
Kubernetes on bare-metal
Bare-metal deployments

- “Many roads lead to Rome”
- Actually a (more or less) solved problem
  - OpenStack
  - Foreman
  - Maas
  - Matchbox
  - ....
What do we need?

- DHCP/proxyDHCP
- PXE/iPXE
- TFTP
- OS image
- OS configuration
- Role-based configuration (e.g. Master/Worker)
Container Linux bare-metal deployment

- Bare-metal deployment with the default toolstack
  - DHCP / TFTP / DNS

- CoreOS Matchbox
  - iPXE Service
  - Deliver assets based on predefined roles

- CoreOS ignition
  - Configure servers on boot-time
How does the Setup work

Source: https://github.com/coreos/matchbox/blob/master/Documentation/network-booting.md#ipxe
Exercise 1: Setup Matchbox + PXE boot

- **What we do:**
  - Setup Wifi/Network (for your laptop)
  - Download: [https://github.com/inovex/fluffy-unicorn](https://github.com/inovex/fluffy-unicorn)
  - Setup Matchbox + terraform
  - Setup DHCP
  - Setup DNS
  - Provision 2 Container Linux nodes
  - Look into the general flow

- **Example is under /exercise_1**
Recap: Kubernetes on bare-metal

- Bare-metal provisioning of OS ✓
- OS configuration with ignition ✓
- Always PXE boot ×
- Role-based configuration ×
- Kubernetes cluster installed ×
- Deployment with a pipeline ×
Challenge always PXE boot

- Matchbox is a pretty simple rendering engine
- Doesn’t hold any state
  - e.g. machine X is provisioned
- How to solve this issue?
  - Any ideas?
Challenge always PXE boot (our solution)

- No access to IPMI
- External DB that holds the state
  - In our case an etcd cluster
- Machine sets its state to “deployed”
- DHCP server decides what to do
  - Provision from Matchbox
  - Boot from disk
- DHCP config is generated with confd
Why etcd?

- Simple to cluster (HA)
- Allows to watch events
- Confd for configuration generation
- Can be used as Vault backend
- ...?
Exercise 2: Always PXE boot

- What we do (additional):
  - Setup etcd
  - Setup confd
  - Generate DHCP config based on Machine state

- Example is under /exercise_2
Recap: Kubernetes on bare-metal

- Bare-metal provisioning of OS ✅
- OS configuration with ignition ✅
- Always PXE boot ✅
- Role-based configuration ❌
- Kubernetes cluster installed ❌
- Deployment with a pipeline ❌
Role-based configuration

- Matchbox groups
- Selectors can be used
  - MAC
  - label (e.g. os=installed)
  - UUID
- Semantic AND (no list support)
  - https://github.com/coreos/matchbox/issues/586
Role-based configuration

```plaintext
// Match machines which have CoreOS Container Linux installed
resource "matchbox_group" "simple-install" {
  name = "simple-install"
  profile = "$\{\text{matchbox_profile.simple.name}\}$"

  selector {
    os = "installed"
  }

  metadata {
    ssh_authorized_key = "$\{\text{var.ssh_authorized_key}\}$"
  }
}
```
Exercise 3: Role-based provisioning

- What we do (additional):
  - Setup two groups
    - based on the MAC
  - Setup write a file on the host (/etc/my_type)
    - MAC -> content
    - 52:54:00:fb:53:a6 -> “I’m a master”
    - 52:54:00:fb:53:a9 -> “I’m a worker”

- Example is under /exercise_3
Recap: Kubernetes on bare-metal

- Bare-metal provisioning of OS ✓
- OS configuration with ignition ✓
- Always PXE boot ✓
- Role-based configuration ✓
- Kubernetes cluster installed ×
- Deployment with a pipeline ×
Kubernetes what do we need?
K8s - what do we need?

- etcd cluster
- At least one Master
- Worker nodes
- Certificates for TLS
  - PKI
  - Or manually generated
- Load Balancer in a multi master setup
  - e.g. https://metallb.universe.tf or HAProxy/Nginx/...
Kubernetes/etcd PKI

- PKI is also needed for the etcd cluster
  - Use TLS everywhere!

- Self-generated certificates
  - Hard to manage the certificates
  - Certificates must be “copied” onto the machines

- Central PKI
  - Auto-generated Certificates
  - Auto-renewal
  - Revocation list

Read: https://jvns.ca/blog/2017/08/05/how-kubernetes-certificates-work
Vault as central PKI

- Simple REST API
- Allows to create multiple CA’s
- Based on policies allow or deny certificate requests
- Can be combined with consul_tempalte
  - Auto-renew certificates (short TTL)
Vault AppRoles

- Authentication mechanism for machines or applications
- Create AppRole ID and SecretID
  - similar to username + password
- Can be used to fetch an Vault token
Big Picture

1. IPAM
2. etcd
3. DHCP/TFTP
4. Vault
5. Node

Asset Mgmt.

Matchbox

confd
Exercise 4: Deploy your K8s cluster

- What we do (additional):
  - Setup Vault
  - Setup consul_template
  - Setup Kubernetes

- Example is under /exercise_4
Recap: Kubernetes on bare-metal

- Bare-metal provisioning of OS ✓
- OS configuration with ignition ✓
- Always PXE boot ✓
- Role-based configuration ✓
- Kubernetes cluster installed ✓
- Deployment with a pipeline ✗
Who knows what CI/CD is?
Continuous integration

- Push into your branch and get Feedback
- Merge changes fast into the master
- Automated build is triggered
- Test run automatically after each push
- Fail fast in the pipeline
- Avoid integration hell → “worked on my machine”
Continuous delivery

- Same as continuous integration
- Allow you to release changes quickly
- Release is completely automated
- Deploy is only clicking one button
Continuous deployment

- Same as continuous delivery
- Every change that passes all stages will be deployed
- No human interaction needed
CI/CD for the infrastructure

- More or less the same as with SE
- Lint/check your configuration files
  - shellcheck / hadolint / ...  
  - kubeval / kubetest
- Test any transformation steps
- Spin up test infrastructure (virtualized)
- Possibility to update your infrastructure by one click
  - Incremental changes have some benefits
Gitlab (CI)

- **Free git repository**
  - Can be self-hosted
- **Integrated CI**
  - Different types of runners
  - Supports building pipelines
  - Specified as a YAML file
Exercise 5: Deploy your K8s cluster over CI/CD

- What we do (additional):
  - Install Gitlab → https://about.gitlab.com/installation/#debian
  - Install a Gitlab runner
  - Create a new Repository and push all files to it
  - Create a dummy pipeline
  - Replace some of the makefile steps

- Example is under /exercise_5
Recap: Kubernetes on bare-metal

- Bare-metal provisioning of OS ✔
- OS configuration with ignition ✔
- Always PXE boot ✔
- Role-based configuration ✔
- Kubernetes cluster installed ✔
- Deployment with a pipeline ✔
Reading list

- https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure
- https://martinfowler.com/bliki/PhoenixServer.html
- https://maas.io
- https://www.theforeman.org
- https://wiki.openstack.org/wiki/Ironic
- https://github.com/coreos/matchbox
- https://cfssl.org
Reading list

- https://blog.digitalocean.com/vault-and-kubernetes
- https://kubernetes.io/docs/concepts/cluster-administration/certificates
- https://github.com/hashicorp/consul-template
- https://about.gitlab.com