Politechnika Warszawska

WYDZIAŁ ELEKTRYCZNY

Instytut Elektrotechniki Teoretycznej
i Systemów Informacyjno-Pomiarowych
Zakład Elektrotechniki Teoretycznej
i Informatyki Stosowanej

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Inżynieria oprogramowania

Implementacja i testy wydajnosci środowiska Kubernetes na maszynach bezdyskowych

Krzysztof Nazarewksi
nr albumu 123456

promotor
mgr inż. Andrzej Toboła

WARSZAWA 2018

Implementacja i testy wydajnosci środowiska Kubernetes na maszynach bezdyskowych

Streszczenie

Celem tej pracy inżynierskiej jest przybliżenie czytelnikowi zagadnień związanych z uruchamianiem systemu Kubernetes na maszynach bezdyskowych.

Zacznę od wyjaśnienia pojęcia systemu bezdyskowego oraz sposobu jego funkcjonowania na przykładzie sieci uczelnianej i wzorującego się na niej przygotowanie przeze mnie lokalnego środowiska.

Następnie opiszę problem izolacji i przydzielania zasobów systemowych na przykładzie wirtualnych maszyn, chroot i konteneryzacji.

W głównej części dokumentu przedstawię pojęcie orkiestrami kontenerami, w jaki sposób odnosi się do wcześniej postawionych problemów. Opiszę alternatywy Kubernetes, jego architekturę oraz sposoby uruchamiania. Na koniec spróbuję uruchomić Kubernetes na maszynach bezdyskowych, problemy z tym związane oraz przedstawię wyniki.

Słowa kluczowe: praca dyplomowa, LaTeX, jakość

Implementing and testing Kubernetes running on diskless machines

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi ac dolor scelerisque, malesuada ex vel, feugiat augue. Suspendisse dictum, elit efficitur vestibulum eleifend, mi neque accumsan velit, at ultricies ex lectus et urna. Pellentesque vel lorem turpis. Donec blandit arcu lacus, vitae dapibus tellus tempus et. Etiam orci libero, mollis in dapibus tempor, rutrum eget magna. Nullam congue libero non velit suscipit, vel cursus elit commodo. Praesent mollis augue quis lorem laoreet, condimentum scelerisque ex pharetra. Sed est ex, gravida a porta in, tristique ac nunc. Nunc at varius sem, sit amet consectetur velit.

Keywords: thesis, LaTeX, quality

WARSZAWA, 1 lutego 1234

POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY

OŚWIADCZENIE

Świadom odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa inżynierska pt. Implementacja i testy wydajnosci środowiska Kubernetes na maszynach bezdyskowych:

- została napisana przeze mnie samodzielnie,
- nie narusza niczyich praw autorskich,
- nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam, że przedłożona do obrony praca dyplomowa nie była wcześniej podstawą postępowania związanego z uzyskaniem dyplomu lub tytułu zawodowego w uczelni wyższej. Jestem świadom, że praca zawiera również rezultaty stanowiące własności intelektualne Politechniki Warszawskiej, które nie mogą być udostępniane innym osobom i instytucjom bez zgody Władz Wydziału Elektrycznego.

Oświadczam ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją elektroniczną.

Krzysztof Nazarewksi.....

Spis treści

Podziękowania

Dziękujemy bardzo serdecznie wszystkim, a w szczególności Rodzinom i Unii Europejskiej...

Zdolny Student i Pracowity Kolega

Rozdział 1

Test

The seminal work (Pizza i in. 2000)

Rozdział 2

Wstęp

Rozdział 3

Systemy bezdyskowe

3.1 Proces uruchamiania maszyny bezdyskowej

Na uruchamianie maszyn bezdyskowych w protokole PXE składają się 3 podstawowe elementy: 1. serwer DHCP, np. isc-dhcp-server lub dnsmasq 2. firmware wspierające PXE, np. iPXE 3. serwer plików (np. TFTP, HTTP, NFS) i/lub sieciowej pamięci masowej (np. iSCSI)

Rozdział 4

Izolacja i przydzielanie zasobów systemowych

4.1 Wirtualizacja

4.2 chroot

4.3 LXC

4.4 Docker, rkt

4.5 LXD

Rozdział 5

Przegląd systemów orkiestracji kontenerami

5.1 Fleet

Fleet[1] jest nakładką na systemd[2] realizująca rozproszony system inicjalizacji systemów. Kontenery są uruchamiane i zarządzane przez systemd.

5.2 Docker Swarm

Docker Swarm[3] jest rozwiązaniem orkiestracji kontenerami od twórców samego Docker'a. Proste w konfiguracji, nie oferuje tak dużych możliwości jak niżej wymienione.

5.3 Kubernetes

Kubernetes[4] jest jednym z najpopularniejszych narzędzi orkiestracji kontenerami. Stworzony przez Google i bazowany na wewnętrznym systemie Borg.

[1] `https://coreos.com/fleet/docs/latest/launching-containers-fleet.html`
[2] `https://www.freedesktop.org/wiki/Software/systemd/`
[3] `https://docs.docker.com/engine/swarm/`
[4] `https://kubernetes.io/`

5.4 Mesos

Apache Mesos[5] zaawansowane narzędzie orkiestracji kontenerami.

5.5 Rancher

Rancher[6] jest platformą zarządzania kontenerami umożliwiającą między innymi zarządzanie klastrem Kubernetes. Od wersji 2.0 twórcy skupiają się na zarządzaniu Kubernetes porzucając inne silniki.

[5] `http://mesos.apache.org/`
[6] `https://rancher.com/`

Rozdział 6

Kubernetes

6.1 Architektura

6.1.1 Komunikacja sieciowa

4 rodzaje sieci: 1. komunikacja wewnątrz Podów (localhost) 2. komunikacja między Podami (SDN/CNI, np. flannel, Calico) 3. komunikacja między Podami i Serwisami (kube-proxy) 4. komunikacja świata z Serwisami

6.1.2 Składowe kontrolujące klaster

- etcd - przechowywanie stanu klastra
- kube-apiserver - interfejs konfiguracyjny klastra (zarówno wewnętrzny jak i zewnętrzny), prowadzi interakcję tylko ze stanem klastra w etcd
- kube-scheduler - proces decydujący na którym węźle klastra uruhamiać Pody (na podstawie dostępnych zasobów, obecnego obciążenia itp.). W skrócie zarządza popytem i podażą na zasoby klastra.
- kube-controller-manager - kontroler klastra dążący do doprowadzenia obecnego stanu klastra do pożądanego na podstawie informacji z kube-apiserver

6.1.3 Składowe workera

- kubelet - monitoruje i kontroluje stan pojedynczego węzła. Na przykład restartuje Pod, który przestał działać na tym samym węźle.
- kube-proxy - proxy i load balancer odpowiedzialny za przekierowanie ruchu do odpowiedniego kontenera
- cAdvisor - monitoruje zużycie zasobów i wydajność kontenerów w ramach jednego węzła

Rozdział 7

Przegląd systemów operacyjnych

Ze względu na obszerność i niejednoznaczność tematu cloud-init rozdział rozpocznę od jego omówienia.

Wszystkie moduły Kubernetes'a są uruchamiane w kontenerach, więc jedynym absolutnie niezbędnym wymaganiem jest uruchamianie Docker'a

Wymaganiami związku z naszym zastosowaniem:

- zdalny dostęp SSH,
- działanie w środowisku bezdyskowym,
- wsparcie narzędzia, którym konfigurujemy Kubernetes

Podstawowe wyznaczniki:

- czy PXE boot działa?
- sposób konfiguracji maszyny
- rozmiar bootowalnego obrazu (kernela i initrd)
- rozmiar minimalnego działającego systemu (z zainstalowanym SSH i Dockerem)
- obsługa NFS/NBD/iSCSI root'a? (zmniejszenie zajmowanego RAMu)

7.1 Konfigurator cloud-init

cloud-init[1] jest standardem oraz implementacją konfiguratora kompatybilnego z wieloma systemami operacyjnymi przeznaczonymi do działania w chmurze.

Standard polega na dostarczeniu pliku konfiguracyjnego w formacie YAML[2] w trakcie lub tuż po inicjalizacji systemu operacyjnego.

[1] `https://cloud-init.io/`

[2] `http://yaml.org/`

7.1.1 Dostępne implementacje

cloud-init

Referencyjny cloud-init zaimplementowany jest w Pythonie, co częściowo tłumaczy duży rozmiar obrazów przeznaczonych dla chmury. Po najmniejszych obrazach Pythona dla Dockera[3] (python:alpine - 89MB i python2:alpine - 72 MB) wnioskuję, że nie istnieje mniejsza dystrybucja Pythona.

```
1 docker pull python:2-alpine > /dev/null
2 docker pull python:alpine > /dev/null
3 docker images | grep alpine
```

Wywiad z developerem cloud-init[4]

coreos-cloudinit

coreos-cloudinit[5] jest częściową implementacją standardu w języku Go przez twórców CoreOS Niestety rok temu przestał być rozwijany[6] i wychodzi z użytku.

RancherOS + coreos-cloudinit

rancher cloud-init[7] jest jest spadkobiercą[8] coreos-cloudinit rozwijanym przez zespół RancherOS, na jego potrzeby.

clr-cloud-init

clr-cloud-init[9] jest wewnętrzną implementacją standardu dla systemu ClearLinux. Powstała z chęci optymalizacji standardu pod ClearLinux oraz pozbycia się zależności referencyjnej implementacji od Python'a.

---

[3] https://hub.docker.com/_/python/

[4] https://www.podcastinit.com/cloud-init-with-scott-moser-episode-126

[5] https://github.com/coreos/coreos-cloudinit

[6] https://github.com/coreos/coreos-cloudinit/commit/3460ca4414fd91de66cd581d997bf453fd895b67

[7] http://rancher.com/docs/os/latest/en/configuration/

[8] https://github.com/rancher/os/commit/e2ed97648ad63455743ebc16080a82ee47f8bb0c

[9] https://clearlinux.org/blogs/announcing-clr-cloud-init

7.2 CoreOS

CoreOS[10] jest pierwszą dystrybucją linuxa dedykowaną zarządzaniu kontenerami. Zawiera dużo narzędzi dedykowanych klastrowaniu i obsłudze kontenerów, w związku z tym zajmuje 342 MB.

- czysta instalacja zajmuje około 600 MB pamięci RAM

7.2.1 Konfiguracja

Konfiguracja przez Container Linux Config[11] transpilowany do Ignition[12]. Transpiler konwertuje ogólną konfigurację na przygotowaną pod konkretne chmury (AWS, GCE, Azure itp.). Dyskwalifikującą wadą tego typu konfiguracji jest brak wsparcia transpilatora dla systemów z rodziny BSD

Poprzednikiem Ignition jest coreos-cloudinit.

7.3 RancherOS

RancherOS[13] jest systemem operacyjnym, w którym tradycyjny system inicjalizacji został zastąpiony trzema poziomami Dockera[14]:

- `bootstrap_docker` działający w initramie, czyli przygotowujący system,
- `system-docker` zastępuje tradycyjnego inita, zarządza wszystkimi programami systemowymi,
- `docker` standardowy docker, interakcja z nim nie może uszkodzić działającego systemu,

Jego głównymi zaletami są mały rozmiar plików startowych (45 MB) oraz prostota konfiguracji.

- zajmuje 700 MB pamięci RAM,

[10] `https://coreos.com/`
[11] `https://coreos.com/os/docs/latest/provisioning.html`
[12] `https://coreos.com/ignition/docs/latest/`
[13] `https://rancher.com/rancher-os/`
[14] `http://rancher.com/docs/os/latest/en/configuration/docker/`

7.3.1 Konfiguracja

RancherOS jest konfigurowany przez własną wersję coreos-cloudinit.

Znaczną przewagą nad oryginałem jest możliwość sekwencyjnego uruchamiania dowolnej ilości plików konfiguracyjnych.

Minimalna konfiguracja pozwalająca na zalogowanie:

```
1 #cloud-config
2 ssh_authorized_keys:
3   - ssh-rsa AAAAB3N...
```

Generuję ją poniższym skryptem na podstawie komendy `ssh-add -L`:

```
1 echo "#cloud-config
2 ssh_authorized_keys:
3 $(ssh-add -L | sed 's/^/  - /g')" > ${cc_dir}/ssh.yml
```

Przydatne jest wyświetlenie kompletnej konfiguracji komendą `ros config export --full`[15].

Niestety aktualnie RancherOS nie zawsze czyta cloud-config[16].

7.4 Project Atomic

Project Atomic[17] jest grupą podobnie skonfigurowanych systemów operacyjnych dedykowaną środowiskom cloud i kontenerom.

Dystrybucje Project Atomic nazywają się Atomic Host, dostępne są następujące warianty:

- Red Hat Atomic Host[18]
- CentOS Atomic Host[19]
- Fedora Atomic Host[20]

7.4.1 Konfiguracja

Atomic Host są konfigurowane systemem cloud-init[21],

[15] https://forums.rancher.com/t/good-cloud-config-reference/5238/3
[16] https://github.com/rancher/os/issues/2204
[17] https://www.projectatomic.io/
[18] https://www.redhat.com/en/resources/enterprise-linux-atomic-host-datasheet
[19] https://wiki.centos.org/SpecialInterestGroup/Atomic/Download/
[20] https://getfedora.org/atomic/download/
[21] https://cloud-init.io/

7.5 Alpine Linux

Alpine Linux[22] jest minimalną dystrybucją Linuxa bazowaną na musl-libc i busybox.

Niestety nie bootuje sie w trybie diskless ze wzgledu na buga, ktorego tworcy nie umieja naprawic.

7.5.1 Konfiguracja

Alpine Backup[23] - spakowane pliki wypakowywane w sekwencji bootu
Alpine Configuration Framework[24]

7.6 ClearLinux

ClearLinux[25]

- "bundle" zamiast pakietów systemowych aktualizowane z całym systemem,
- skoncentrowany na wydajności na procesorach Intel,
- skąpa i trudna w nawigacji dokumentacja systemu,
- lokalizacja wszystkich modyfikacji w /var i /etc (prosty reset),
- instalacja samego docker'a + serwera ssh zajmuje 700 MB w RAMie

7.6.1 Linki

- https://www.infoworld.com/article/3159658/linux/6-key-points-about-intels-hot-new-linux-distro.html

[22] `https://alpinelinux.org/`
[23] `https://wiki.alpinelinux.org/wiki/Alpine_local_backup`
[24] `http://wiki.alpinelinux.org/wiki/Alpine_Configuration_Framework_Design`
[25] `https://clearlinux.org/`

Rozdział 8

Przegląd sposobów konfiguracji klastra Kubernetes

Najpopularniejszym rozwiązaniem konfiguracji klastra Kubernetes jest kops[1], ale jak większoś

Interesują nas tylko i wyłącznie rozwiązania bare-metal:

- kubeadm[2]
  - Install with kubadm[3]
- kubespray[4]
  - zestaw skryptów Ansible konfigurujących klaster na jednym z wielu systemów operacyjnych
  - dąży do zostania tzw. Operatorem[5] korzystającym z kubeadm
- Fedora via Ansible[6]
  - deprekowane na rzecz kubespray - odpada
- Rancher 2.0[7], korzysta z RKE

[1] https://github.com/kubernetes/kops

[2] https://kubernetes.io/docs/reference/setup-tools/kubeadm/kubeadm/

[3] https://kubernetes.io/docs/setup/independent/install-kubeadm/

[4] https://github.com/kubernetes-incubator/kubespray

[5] https://github.com/kubernetes-incubator/kubespray/blob/master/docs/comparisons.md

[6] https://kubernetes.io/docs/getting-started-guides/fedora/fedora_ansible_config/

[7] http://rancher.com/rancher2-0/

- Rancher Kubernetes Installer[8]
- Rancher 1.X[9]

8.1 Rancher 1.X/2.0

Wygodne narzędzie do uruchamiania i monitorowania klastra Kubernetes, ale wymaga interakcji użytkownika.

```
#rancher_version=latest
rancher_version=v2.0.0-alpha10
docker run --rm --name rancher -d -p 8080:8080 rancher/server:$
    ↪ {rancher_version}
```

Najpierw należy zalogować się do panelu administracyjnego Ranchera i przeprowadzić podstawową konfigurację (adres Ranchera + uzyskanie komendy).

Następnie w celu dodania węzła do klastra wystarczy wywołać jedną komendę udostępnioną w panelu administracyjnym Ranchera na docelowym węźle, jej domyślny format to:

```
wersja_agenta=v1.2.9
ip_ranchera=192.168.56.1
skrypt=B52944BEFAA613F0CE90:1514678400000:
    ↪ E2yB6KfxzSix4YHti39BTw5RbKw

sudo docker run --rm --privileged \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v /var/lib/rancher:/var/lib/rancher \
  rancher/agent:${wersja_agenta} \
  http://${ip_ranchera}:8080/v1/scripts/${skrypt}
```

W ciągu 2 godzin przeglądu nie udało mi się zautomatyzować procesu uzyskiwania ww. komendy.

Następnie w cloud-config'u RancherOS'a możemy dodać ww. komendę w formie:

```
#cloud-config
runcmd:
- docker run --rm --privileged -v /var/run/docker.sock:/var/run
    ↪ /docker.sock -v /var/lib/rancher:/var/lib/rancher
```

[8] http://rancher.com/announcing-rke-lightweight-kubernetes-installer/
[9] https://rancher.com/rancher/

```
    ↪ rancher/agent:v1.2.9 http://192.168.56.1:8080/v1/scripts
    ↪ /...
```

8.2 kubespray-cli

Jest to narzędzie ułatwiające korzystanie z `kubespray`. Z powodu błędu[10] logiki narzędzie nie radzi sobie z brakiem Python'a na domyślnej dystrybucji CoreOS'a, mimo że sam `kubespray` radzi sobie z nim świetnie. Do uruchomienia na tym systemie potrzebne jest ręczne wywołanie roli `bootstrap-os`[11] z `kubespray` zanim przystąpimy do właściwego deploy'u.

```
1  #!/usr/bin/env bash
2  set -e
3
4  #pip2 install ansible kubespray
5  get_coreos_nodes() {
6    for node in $@
7    do
8      echo -n node1[
9      echo -n ansible_host=${node},
10     echo -n bootstrap_os=coreos,
11     echo -n ansible_user=core,
12     echo -n ansible_default_ipv4.address=${node}
13     echo ]
14   done
15 }
16
17 NODES=($(get_coreos_nodes 192.168.56.{10,12,13}))
18 echo NODES=${NODES[@]}
19 kubespray prepare -y --nodes ${NODES[@]}
20 cat > ~/.kubespray/bootstrap-os.yml << EOF
21 - hosts: all
22   become: yes
23   gather_facts: False
24   roles:
25   - bootstrap-os
26 EOF
```

[10] https://github.com/kubespray/kubespray-cli/issues/120

[11] https://github.com/kubernetes-incubator/kubespray/blob/master/roles/bootstrap-os/tasks/main.yml

```
27
28 (cd ~/.kubespray; ansible-playbook -i inventory/inventory.cfg
   ↪ bootstrap-os.yml)
29 kubespray deploy -y --coreos
```

Wykrzacza się na kroku czekania na uruchomienie `etcd` ponieważ oczekuje połączenia na NATowym interfejsie z adresem `10.0.3.15` zamiast host network z adresem `192.168.56.10`, stąd `ansible_default_ipv4.address`.

Według użytkowników oficjalnego Slacka kubespray[12] `kubespray-cli` jest deprekowane.

8.3 kubespray

Kod znajduje się w moim repozytorium kubernetes-cluster[13].

```
1  #!/usr/bin/env bash
2  cd $(dirname "$(readlink -f "$0")")
3  source ./setup-cluster-vars
4  cd ${dir}
5
6   based on https://github.com/kubernetes-incubator/kubespray/
       ↪ blob/master/docs/getting-started.md#building-your-own-
       ↪ inventory
7  cp -r inventory -T ${inventory}
8  python3 contrib/inventory_builder/inventory.py ${IPS[@]}
9
10 cat > ${inventory}/group_vars/all.yml << EOF
11 bootstrap_os: coreos
12 #loadbalancer_apiserver:
13 #  address: 0.0.0.0
14 #  port: 8080
15 kube_basic_auth: true
16 kubeconfig_localhost: true
17 kubectl_localhost: true
18 download_run_once: True
19 EOF
20
21 ansible-playbook -i ${inventory}/inventory.cfg cluster.yml -b -
       ↪ v
```

[12] https://kubernetes.slack.com/messages/kubespray

[13] https://github.com/nazarewk/kubernetes-cluster

```
22
23 cd ${dir}/artifacts
24
25 echo Staring kubectl proxy
26 echo http://localhost:8001/api/v1/namespaces/kube-system/
    ↪ services/https:kubernetes-dashboard:/proxy/#!/login
27 ./kubectl proxy
```

8.3.1 Dostęp do dashboard'u

https://github.com/kubernetes/dashboard/wiki/Access-control#kubeconfig
https://github.com/kubernetes-incubator/kubespray/blob/master/docs/getting-started.md#accessing-kubernetes-dashboard

8.3.2 Napotkane błędy

Błąd przy ustawieniu `loadbalancer_apiserver.address` na `0.0.0.0`:

```
1 TASK [kubernetes-apps/cluster_roles : Apply workaround to allow
    ↪  all nodes with cert O=system:nodes to register]
    ↪ ***********************************
2 Wednesday 17 January 2018  22:22:59 +0100 (0:00:00.626)
    ↪ 0:08:31.946 *****
3 fatal: [node2]: FAILED! => {"changed": false, "msg": "error
    ↪ running kubectl (/opt/bin/kubectl apply --force --
    ↪ filename=/etc/kubernetes/node-crb.yml) command (rc=1):
    ↪ Unable to connect to the server: http: server gave HTTP
    ↪ response to HTTPS client\n"}
4 fatal: [node1]: FAILED! => {"changed": false, "msg": "error
    ↪ running kubectl (/opt/bin/kubectl apply --force --
    ↪ filename=/etc/kubernetes/node-crb.yml) command (rc=1):
    ↪ Unable to connect to the server: http: server gave HTTP
    ↪ response to HTTPS client\n"}
```

Rozdział 9

Q&A

9.1 Czy wszystko zawsze trzeba sciagac z netu - nie mozna z lokalnego serwera?

Można zestawić lokalny rejestr Dockera[1] jako proxy cachujące[2].

9.2 Jak zachować stan bezdyskowego RancherOS'a?

Jedynym narzędziem do "zachowywania stanu" bezdyskowego Ranchera i praktycznie wszystkich cloudowych systemów uruchamianych bez dysku jest cloud-init.

Normalnie konfigurowany jest przez własny cloud-init, aktualnie nie zawsze działa ze względu na bugi.

9.3 Co musi zawierac cloud-config dla serwera a co dla agentow?

Sam RancherOS nie zarządza kontenerami, do tego potrzebne jest uruchomienie serwera Ranchera.

[1] `https://docs.docker.com/registry/`

[2] `https://docs.docker.com/registry/recipes/mirror/`

Bibliografia

Pizza, Mariagrazia, Vincenzo Scarlato, Vega Masignani, Marzia Monica Giuliani, Beatrice Arico, Maurizio Comanducci, Gary T Jennings, i in. 2000. „Identification of vaccine candidates against serogroup B meningococcus by whole-genome sequencing". *Science* 287 (5459). American Association for the Advancement of Science:1816–20.