

Devops examen 2017

Har valt att dela upp min lösning i mindre steg, startade med ett basuppsätt med grundläggande funktionalitet där ändå stora delar är manuellt konfigurerade.

I detta dokument kommer jag beskriva processen steg för steg, i tillägg har jag spelat in 2 screencasts som visar mina 2 färdiga projekt.

Allt jag använt för att sätta upp mina projekt finns under följande repo:

https://github.com/josoder/devops_exam

1. Initial:

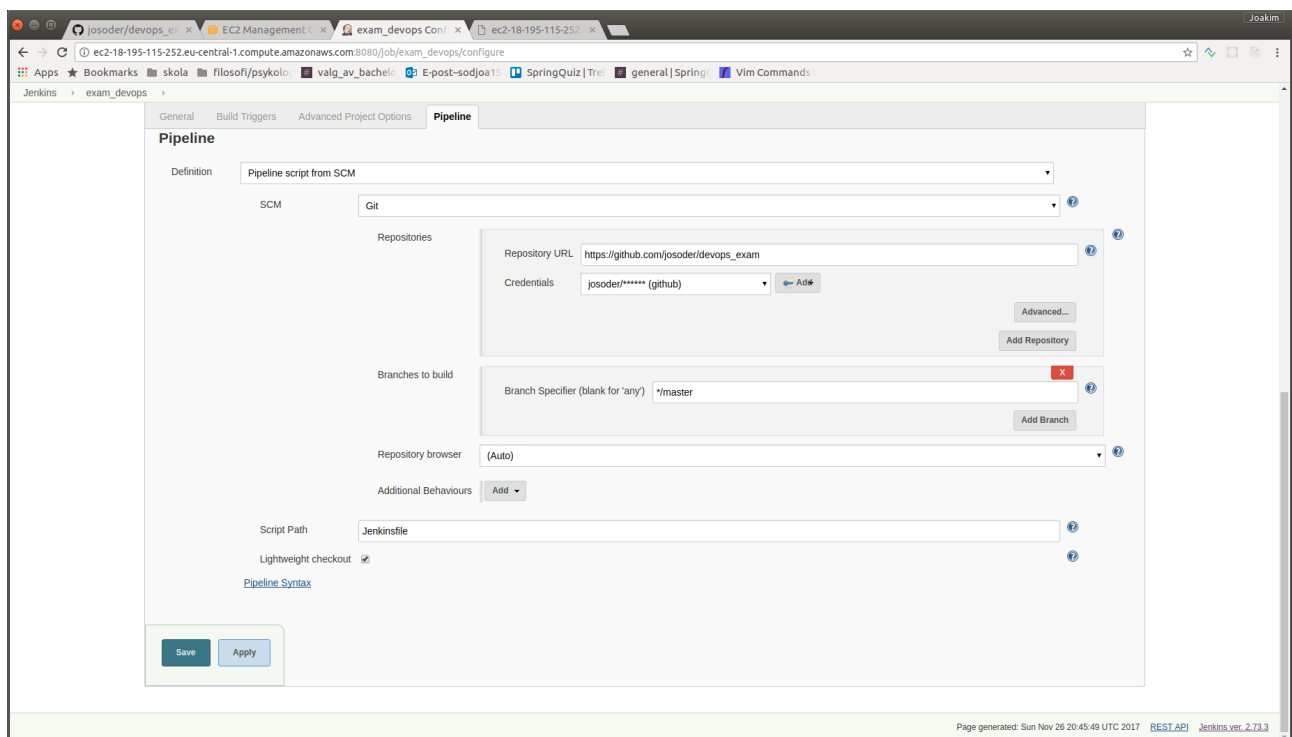
https://github.com/josoder/devops_exam/tree/f51deb18f4272f571358a716529429fc305d83a6

Manuellt uppsatt Jenkins-server som bygger min app som körs i en AWS ec2 instans.

Appen och jenkinsfilen finns i mitt git-repo, min pipeline laddas från git-repo och utför som första steg en kloning av hela mitt repo.

Applikationen byggs i en docker image med hjälp av docker-filen som också den ligger på git.

Efter att en image har byggts så utförs en test(som just nu inte testar något relevant för appen men finns där för att demonstrera vart tester skall utföras), före min nya image publiceras till dockerhub(https://hub.docker.com/r/josoder/devops_docker/). Det sista steget 'publish' startar till slut min nya image i en container på byggservern som blivit konfigurerad med port-forwarding(5000:5000) och appen är därmed offentligt synlig på nätet.



devops_exam/Jen... x EC2 Management... x jenkins / exam_dev... x React App x josoder/devops_ei... x

ec2-18-195-115-252.eu-central-1.compute.amazonaws.com:8080/blue/organizations/jenkins/exam_devops/detail/exam_devops/8/pipeline/43

Apps ★ Bookmarks skola filosof/psykolo valg_av_bachelo E-post-sodjoe15 SpringQuiz | Tre general | Spring Vim Commands

✓ exam_devops 8 Pipeline Changes Tests Artifacts ↺ ⚙️ 📄 Logout ✕

Branch: — 19s No changes
Commit: — 17 minutes ago Started by user Joakim Söderstrand

Start clone build test push image publish End


Steps publish

✓ > docker run -d -i -p 5000:5000 josoder/devops_docker -- Shell Script <1s

devops_exam/Jen... x EC2 Management... x exam_devops Con... x React App x josoder/devops_ei... x

ec2-18-195-115-252.eu-central-1.compute.amazonaws.com:5000

Apps ★ Bookmarks skola filosof/psykolo valg_av_bachelo E-post-sodjoe15 SpringQuiz | Tre general | Spring Vim Commands



Welcome to React

To get started, edit `src/App.js` and save to reload.

2. Automatiserat uppsätt av Jenkins och ec2 instans

https://github.com/josoder/devops_exam/tree/83122d36bd249e29535118279a905a198525d608

I steg 2 har jag valt att automatisera uppsättningen av Jenkins och ec2 instansen med hjälp av ansible.

För detta krävdes installation av ansible och boto(<https://pypi.python.org/pypi/boto>, pakethanterare som erbjuder ett gränssnitt mot AWS) lokalt. Behövde också sätta 2 miljövariabler lokalt för att få root-access till mitt amazon konoto.

```
export AWS_ACCESS_KEY_ID=''
export AWS_SECRET_ACCESS_KEY=''
```

```
josoder@josoder:~/devops/devops_exam/ansible$ ansible-playbook -i ./hosts create-ec2.yml

PLAY [Create an EC2 Instance] *****

TASK [Create a security group] *****
changed: [localhost -> localhost]

TASK [Launch the new EC2 Instance] *****
changed: [localhost -> localhost]

TASK [Add the newly created EC2 instance(s) to the local host group (located inside the directory)] ***
changed: [localhost -> localhost] => (item={u'kernel': None, u'root_device_type': u'efs', u'private_dns_name': u'ip-172-31-39-227.eu-central-1.compute.internal', u'public_ip': u'18.195.40.173', u'private_ip': u'172.31.39.227', u'id': u'i-0cf81c1ef830d1c3d', u'efs_optimized': False, u'state': u'running', u'virtualization_type': u'hvm', u'architecture': u'x86_64', u'ramdisk': None, u'block_device_mapping': {u'/dev/sda1': {u'status': u'attached', u'delete_on_termination': True, u'volume_id': u'vol-0dd5d04e9d8453022'}}, u'key_name': u'exam-jenkins', u'image_id': u'ami-97e953f8', u'tenancy': u'default', u'groups': {u'sg-4f188325': u'jenkins-buildserver'}, u'public_dns_name': u'ec2-18-195-40-173.eu-central-1.compute.amazonaws.com', u'state_code': 16, u'tags': {}, u'placement': u'eu-central-1b', u'ami_launch_index': u'0', u'dns_name': u'ec2-18-195-40-173.eu-central-1.compute.amazonaws.com', u'region': u'eu-central-1', u'launch_time': u'2017-11-27T15:51:36.000Z', u'instance_type': u't2.micro', u'root_device_name': u'/dev/sda1', u'hypervisor': u'xen'})

TASK [Wait for SSH to come up] *****
ok: [localhost -> localhost] => (item={u'kernel': None, u'root_device_type': u'efs', u'private_dns_name': u'ip-172-31-39-227.eu-central-1.compute.internal', u'public_ip': u'18.195.40.173', u'private_ip': u'172.31.39.227', u'id': u'i-0cf81c1ef830d1c3d', u'efs_optimized': False, u'state': u'running', u'virtualization_type': u'hvm', u'architecture': u'x86_64', u'ramdisk': None, u'block_device_mapping': {u'/dev/sda1': {u'status': u'attached', u'delete_on_termination': True, u'volume_id': u'vol-0dd5d04e9d8453022'}}, u'key_name': u'exam-jenkins', u'image_id': u'ami-97e953f8', u'tenancy': u'default', u'groups': {u'sg-4f188325': u'jenkins-buildserver'}, u'public_dns_name': u'ec2-18-195-40-173.eu-central-1.compute.amazonaws.com', u'state_code': 16, u'tags': {}, u'placement': u'eu-central-1b', u'ami_launch_index': u'0', u'dns_name': u'ec2-18-195-40-173.eu-central-1.compute.amazonaws.com', u'region': u'eu-central-1', u'launch_time': u'2017-11-27T15:51:36.000Z', u'instance_type': u't2.micro', u'root_device_name': u'/dev/sda1', u'hypervisor': u'xen'})

TASK [Add tag to Instance(s)] *****
changed: [localhost -> localhost] => (item={u'kernel': None, u'root_device_type': u'efs', u'private_dns_name': u'ip-172-31-39-227.eu-central-1.compute.internal', u'public_ip': u'18.195.40.173', u'private_ip': u'172.31.39.227', u'id': u'i-0cf81c1ef830d1c3d', u'efs_optimized': False, u'state': u'running', u'virtualization_type': u'hvm', u'architecture': u'x86_64', u'ramdisk': None, u'block_device_mapping': {u'/dev/sda1': {u'status': u'attached', u'delete_on_termination': True, u'volume_id': u'vol-0dd5d04e9d8453022'}}, u'key_name': u'exam-jenkins', u'image_id': u'ami-97e953f8', u'tenancy': u'default', u'groups': {u'sg-4f188325': u'jenkins-buildserver'}, u'public_dns_name': u'ec2-18-195-40-173.eu-central-1.compute.amazonaws.com', u'state_code': 16, u'tags': {}, u'placement': u'eu-central-1b', u'ami_launch_index': u'0', u'dns_name': u'ec2-18-195-40-173.eu-central-1.compute.amazonaws.com', u'region': u'eu-central-1', u'launch_time': u'2017-11-27T15:51:36.000Z', u'instance_type': u't2.micro', u'root_device_name': u'/dev/sda1', u'hypervisor': u'xen'})

PLAY RECAP *****
localhost                : ok=5    changed=4    unreachable=0    failed=0

josoder@josoder:~/devops/devops_exam/ansible$
```

The screenshot shows the AWS Management Console interface. The top navigation bar includes the AWS logo and various service links. The left sidebar contains a navigation menu with categories like EC2 Dashboard, ELASTIC BLOCK STORE, NETWORK & SECURITY, and more. The main content area displays the 'Instances' tab, which lists EC2 instances in a table. The table has columns for Name, Instance ID, Instance Type, Availability Zone, Instance State, Status Checks, Alarm Status, Public DNS (IPv4), IPv4 Public IP, IPv6 IPs, Key Name, and Monitoring. One instance, 'jenkinsserver', is highlighted in blue and is in a 'running' state. Below the table, the details for the selected instance are shown, including its ID, state, type, and various IP addresses.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name	Monitoring
jenkinsserver	i-0cf81c1ef830d1c3d	t2.micro	eu-central-1b	running	2/2 checks ...	None	ec2-18-195-40-173.eu-...	18.195.40.173	-	exam-jenkins	disabled

Instance: i-0cf81c1ef830d1c3d (jenkinsserver) Public DNS: ec2-18-195-40-173.eu-central-1.compute.amazonaws.com

Description	Status Checks	Monitoring	Tags
Instance ID	i-0cf81c1ef830d1c3d	Public DNS (IPv4)	ec2-18-195-40-173.eu-central-1.compute.amazonaws.com
Instance state	running	IPv4 Public IP	18.195.40.173
Instance type	t2.micro	IPv6 IPs	-
Elastic IPs	-	Private DNS	ip-172-31-39-227.eu-central-1.compute.internal
Availability zone	eu-central-1b	Private IPs	172.31.39.227
Security groups	jenkins-buildserver - view inbound rules	Secondary private IPs	-
Scheduled events	No scheduled events	VPC ID	vpc-8f2e30e7

Det gick fint så långt men fick sedan problem när jag försökte pinga med ansible, först med autentisering. Löste det till slut genom att lägga till en privat nyckel .pem fil med ssh-agent → ssh-add nyckel.pem.

Då uppstod nästa problem:

```
<18.194.229.189> (0, '/bin/sh: 1: /usr/bin/python: not found\r\n', 'Shared connection to 18.194.229.189 closed.\r\n')
```

Amazons ec2 instanser(ubuntu 16.04) kommer inte med en brukbar python version förinstallerad, det första jag gör i den playbook'en som sköter jenkins installationen efter att min ec2 instans är upprättad är därmed att installera python. För att få till att köra en playbook mot instansen fick jag också lägga till en paramter till min inventory fil för att specificera ansible_python_interpreter.

```

jtosoder@jtosoder: /devops/devops_exam/ansible/ansible$ ansible -n ping jenkinsserver -vvv
ansible 2.4.1.0
config file = /home/jtosoder/devops/devops_exam/ansible/ansible/ansible.cfg
configured module search path = [u '/home/jtosoder/.ansible/plugins/modules', u '/usr/share/ansible/plugins/modules']
ansible python module location = /usr/lib/python2.7/dist-packages/ansible
executable location = /usr/bin/ansible
python version = 2.7.12 (default, Nov 19 2016, 00:48:10) [GCC 5.4.0 20160609]
using /home/jtosoder/devops/devops_exam/ansible/ansible/ansible.cfg as config file
Parsed /home/jtosoder/devops/devops_exam/ansible/ansible/inventory source with int plugin
META: ran handlers
Using module file /usr/lib/python2.7/dist-packages/ansible/modules/system/ping.py
[18.194.229.189]: ESTABLISH SSH CONNECTION FOR USER: ubuntu
[18.194.229.189]: SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthentication=no -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey -o PasswordAut
hentication=no -o User=ubuntu -o ConnectTimeout=10 -o ControlPath=/home/jtosoder/.ansible/cp/3b1b829732 18.194.229.189 '/bin/sh -c \''''echo - && sleep 0\''''''
[18.194.229.189]: (0, ' /home/ubuntu\n', '')
[18.194.229.189]: ESTABLISH SSH CONNECTION FOR USER: ubuntu
[18.194.229.189]: SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthentication=no -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey -o PasswordAut
hentication=no -o User=ubuntu -o ConnectTimeout=10 -o ControlPath=/home/jtosoder/.ansible/cp/3b1b829732 18.194.229.189 '/bin/sh -c \''''( (unask 77 && nckdr -p -' echo /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682
1511811345.85-38333606820682 "' && echo ansible-tmp-1511811345.85-38333606820682"' echo /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682/ping.py
[18.194.229.189]: (0, ' /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682/ping.py
[18.194.229.189]: SSH: EXEC sftp -b -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthentication=no -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey -o Passw
ordAuthentication=no -o User=ubuntu -o ConnectTimeout=10 -o ControlPath=/home/jtosoder/.ansible/cp/3b1b829732 [18.194.229.189]
[18.194.229.189]: (0, 'sftp put -C /tmp/tmp059Jkd /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682/ping.py\n', '')
[18.194.229.189]: ESTABLISH SSH CONNECTION FOR USER: ubuntu
[18.194.229.189]: SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthentication=no -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey -o PasswordAut
hentication=no -o User=ubuntu -o ConnectTimeout=10 -o ControlPath=/home/jtosoder/.ansible/cp/3b1b829732 18.194.229.189 '/bin/sh -c \''''chmod u+x /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-3833360
6820682 /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682/ping.py && sleep 0\''''''
[18.194.229.189]: (0, ' /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682/ping.py && sleep 0\n', '')
[18.194.229.189]: ESTABLISH SSH CONNECTION FOR USER: ubuntu
[18.194.229.189]: SSH: EXEC ssh -C -o ControlMaster=auto -o ControlPersist=60s -o KbdInteractiveAuthentication=no -o PreferredAuthentications=gssapi-with-mic,gssapi-keyex,hostbased,publickey -o PasswordAut
hentication=no -o User=ubuntu -o ConnectTimeout=10 -o ControlPath=/home/jtosoder/.ansible/cp/3b1b829732 -t 18.194.229.189 '/bin/sh -c \''''/usr/bin/python /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682/ping.py; rm -rf /home/ubuntu/.ansible/tmp/ansible-tmp-1511811345.85-38333606820682 && sleep 0\''''''
[18.194.229.189]: (0, '\r\n[Invocation]: {"module_args": {"data": "pong"}, "ping": "pong"}\r\n', 'Shared connection to 18.194.229.189 closed.\r\n')
[18.194.229.189] | SUCCESS =>
  "changed": false,
  "failed": false,
  "invocation": {
    "module_args": {
      "data": "pong"
    }
  },
  "ping": "pong"
}
META: ran handlers
META: ran handlers
jtosoder@jtosoder: /devops/devops_exam/ansible/ansible$

```

För att installera jenkins använde jag mig av "geerlingguy.jenkins" som är en "role" som finns tillgänglig på ansible-galaxy. Denna process gick till skillnad från själva installationen av instansen väldigt smidigt och fungerade på första försöket:

```

[INFO] org.apache.commons.httpclient.HttpMethodPost: You are authenticated as: anonymous, *X-You-Are-In-Group: Disabled, jenkins-99462: use -Dmudson.security.AccessDeniedException2.REPORT
GROUP_HEADER=Strme or use /whoAmI to diagnose, *X-Requested-Permission: hudson.model.Hudson.Read, *X-Permission-Imply-By: hudson.security.Permission.GenericRead, *X-Permission-Imply: hudson
model.Hudson.Administrator, "Content-Length: 830", "Server: Jetty(9.4.z-SNAPSHOT)", "...", "<DOCTYPE html><html><head><meta http-equiv='refresh' content='1;url=/login?from=X2FcclnX2?'>/>
<script>window.location.replace('/login?from=X2FcclnX2?')</script></head><body style='background-color:white; color:white;'>...", "Authentication required", "<!--, 'You are authenticated as: anonymo
us', '<script>that you are in the '... Permission you need to have (but didn't): hudson.model.Hudson.Read, ... which is implied by: hudson.security.Permission.GenericRead, ... which is implied by: h
udson.model.Hudson.Administrat*, ..., </body></html>"

TASK [geerlingguy.jenkins : Get the jenkins-cli.jarfile from the Jenkins server.] *****
changed: [18.194.229.189] => ("attempts": 1, "changed": true, "checksum_dest": null, "checksum_src": "af4c26bb49ef13af3bb3926949607f8f15d33ddad", "dest": "/opt/jenkins-cli.jar", "failed": false, "gid": 0,
"owner": "root", "md5sum": "6766621ce5e8addf3f80e315d0dfaz2", "mode": "0644", "msg": "OK (3234932 bytes)", "owner": "root", "size": 3234932, "src": "/tmp/tmpPzbtPTl", "state": "file", "status_code": 200,
"uid": 0, "url": "http://localhost:8080/jnlpJars/jenkins-cli.jar")

TASK [geerlingguy.jenkins : Remove Jenkins security init scripts after first startup.] *****
changed: [18.194.229.189] => ("changed": true, "failed": false, "path": "/var/lib/jenkins/init.groovy.d/basic-security.groovy", "state": "absent")

TASK [geerlingguy.jenkins : Get Jenkins admin password from file.] *****
skipping: [18.194.229.189] => ("censored": "the output has been hidden due to the fact that 'no_log: true' was specified for this result")

TASK [geerlingguy.jenkins : Set Jenkins admin password fact.] *****
ok: [18.194.229.189] => ("censored": "the output has been hidden due to the fact that 'no_log: true' was specified for this result")

TASK [geerlingguy.jenkins : Get Jenkins admin token from file.] *****
skipping: [18.194.229.189] => ("censored": "the output has been hidden due to the fact that 'no_log: true' was specified for this result")

TASK [geerlingguy.jenkins : Set Jenkins admin token fact.] *****
ok: [18.194.229.189] => ("censored": "the output has been hidden due to the fact that 'no_log: true' was specified for this result")

TASK [geerlingguy.jenkins : Create update directory] *****
ok: [18.194.229.189] => ("changed": false, "failed": false, "gid": 110, "group": "jenkins", "mode": "0755", "owner": "jenkins", "path": "/var/lib/jenkins/updates", "size": 4096, "state": "directory", "uid": 112)

TASK [geerlingguy.jenkins : Download current plugin updates from Jenkins update site.] *****
changed: [18.194.229.189] => ("changed": true, "dest": "/var/lib/jenkins/updates/default.json", "failed": false, "gid": 110, "group": "jenkins", "mode": "0440", "msg": "file already exists but file attrib
utes changed", "owner": "jenkins", "size": 1277966, "state": "file", "uid": 112, "url": "http://updates.jenkins-ci.org/update-center.json")

TASK [geerlingguy.jenkins : Remove first and last line from json file] *****
ok: [18.194.229.189] => ("changed": false, "failed": false, "msg": "")

TASK [geerlingguy.jenkins : Install Jenkins plugins using password.] *****

TASK [geerlingguy.jenkins : Install Jenkins plugins using token.] *****

PLAY RECAP *****
18.194.229.189      : ok=31   changed=13   unreachable=0    failed=0

```

För docker installationen använde jag "geerlingguy.docker", med en liten modifikation i /tasks/main där jag lägger till jenkins user i docker gruppen.

Med instansen uppe så var det bara att återupprätta min pipeline, lägga till credentials för git och dockerhub och installera ett par plugins. Bortsett från dessa sista steg så är hela uppsättet av byggservern nu automatiserat med hjälp av ansible.

Valde att dela upp skapning av ec2 instansen och installationen av jenkins på den nyupprättade instansen i 2 playbooks, create-ec2.yml och install_jenkins.yml.

All konfiguration till ansible finns på mitt github-repo under /ansible.

https://github.com/josoder/devops_exam/tree/master/ansible/jenkins_setup

3. Docker swarm

https://github.com/josoder/devops_exam/commit/f76437b585a5a9b4198ff8d7a7a2e414b39a82aa

Istället för att köra appen på samma instans som jenkins så satte jag upp en docker-swarm på mitt aws-konto. Jag kan då skallera appen när den blir satt i produktion och ha den körande som en service vilket gör den betydligt mycket mer robust.

Tänkte först försöka mig på att installera min swarm med ansible men insåg snabbt att det var mer komplicerat än jag hade trott. Jag bestämde mig därför för att byta strategi, undersökte mina möjligheter och upptäckte att det fanns möjlighet att skapa ett docker-swarm kluster med en template i aws, så kallat cloudformation-template. Se länk nedanför.

<https://docs.docker.com/docker-for-aws/#docker-community-edition-ce-for-aws>

```
josoder@josoder:~$ docker run --rm -ti -v /var/run/docker.sock:/var/run/docker.sock -e DOCKER_HOST dockercloud/client josoder/devops-exam
Unable to find image 'dockercloud/client:latest' locally
latest: Pulling from dockercloud/client
90f4dba627d6: Pull complete
9ce3e514b7fa: Pull complete
ca5f3b3b0b8e: Pull complete
19aaa36bba7a: Pull complete
Digest: sha256:82aecd212932860780485503ec5407c337aba33b0baa0a2f82adaae73eb1cc7
Status: Downloaded newer image for dockercloud/client:latest
Use your Docker ID credentials to authenticate:
Username: josoder
Password:

=> You can now start using the swarm josoder/devops-exam by executing:
    export DOCKER_HOST=tcp://127.0.0.1:32768
josoder@josoder:~$ export DOCKER_HOST=tcp://127.0.0.1:32768
josoder@josoder:~$ docker node ls
ID                                HOSTNAME                                STATUS    AVAILABILITY    MANAGER STATUS
ewp62bxankat1589wkyxmawf4        ip-172-31-2-123.eu-central-1.compute.internal    Ready    Active
25oi12jumvyjm9k8kxnd3kefu *      ip-172-31-22-57.eu-central-1.compute.internal    Ready    Active
g4z6rksx2qi6elofvfa8rcy7l        ip-172-31-28-160.eu-central-1.compute.internal    Ready    Active
```

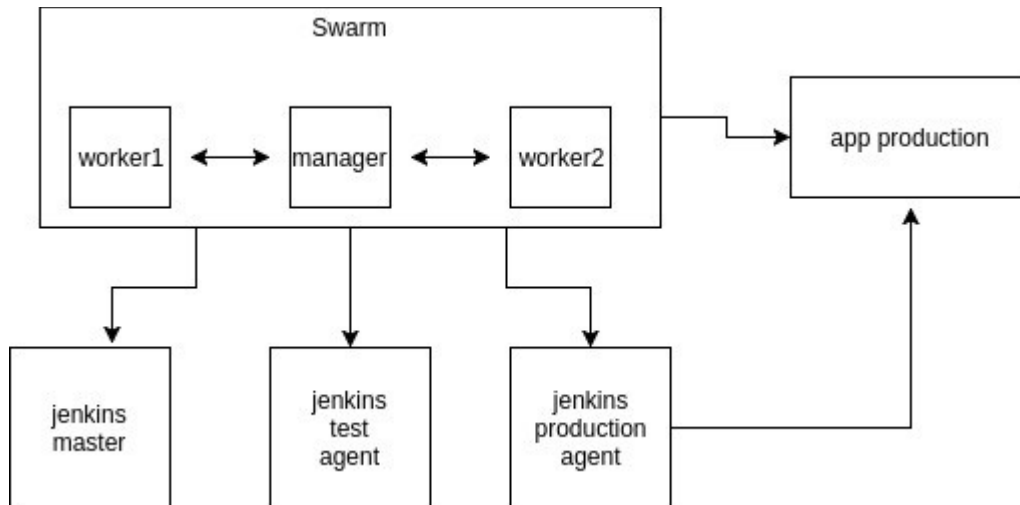
För att få tillgång till min manger-nod i mitt nyskapade kluster från jenkins använde jag ett plugin till jenkins som passande nog heter ssh-agent och gör precis det som namnet antyder.

Nu ansluter jag till mitt kluster istället via ssh i min jenkins pipeline och startar min app som en service. Vilket gör att den kommer vara mer robust och jag har möjlighet till att skallera den. Blev dock inte nöjd med mitt uppsätt. Jag har bara en instans av jenkins utan slaves så om den skulle krascha så får jag problem.

Vill också ha möjligheten för att köra jenkins som service.

4. Jenkins i docker swarm

https://github.com/josoder/devops_exam/tree/master/swarmv2



Eftersom att jag inte blev helt nöjd med mitt förra steg bestämde jag för att ge mig på att köra min jenkins server i min docker swarm. Alltså som en docker service(stack). För detta skapade jag en ny mapp på mitt repo och byggde detta steg som ett separat projekt.

Jag valde att bygga en ny jenkins-image med hjälp av <https://github.com/vfarcic/docker-flow-stacks/tree/master/jenkins>, denna nya image är helt automatisk utan setup-wizard och jag kan med hjälp av docker-secret skapa användaren till jenkins utan att manuellt behöva göra detta när jenkins startar första gången. Denna image installerar dessutom en mängd plugins för mig, så jag slipper göra det manuellt.

Nytt för denna version är också att jag fixat en github-hook som gör att jenkins automatiskt kommer att bygga vid push och att jag i tillägg till jenkins-master stacken kör 2 jenkins-agents som användes till mina olika steg vid bygg, en för test och för produktion.

Pros:

Genom att köra jenkins i en swarm är den nu feltolerant, den körs nu som en service, om jenkins av en eller annan grund skulle krascha så kommer en ny instans att startas direkt och vara uppe på några minuter, helt automatiskt. Samma med appen när den blir satt i produktion.

Har även satt upp min stack så att jenkins körs på en shared, named volume och kommer därmed inte heller att tappa state.

Har separerat test och produktionsmiljö.

Allt körs i kluster med möjligheten för att skalera.

All känslig information som används blir skapade som docker-secrets vilket betyder att den kommer vara krypterad under hela processen och därmed vara säkra(re).

Agents:

I min swarm kör jag en jenkins master stack, en jenkins agent stack för test och en jenkins agent stack för produktion. Alla dessa är startade med hjälp av compose-filer och finns tillgängliga under mitt repo på länken ovanför(under rubriken).

Cons:

Sättet jag har satt upp det nu är för att demonstrera möjligheterna i produktion hade man hellre kört med flera olika kluster, alltså ett kluster för master, 1 för test och 1 för produktion. Men det kostar pengar om man skall köra det i cloud, så jag nöjde mig med 1 kluster där jag gör allting.

Att sätta upp jenkins på detta sättet var betydligt mer avancerat än jag hade räknat med och tog lång tid att få till. Stötte på många problem på vägen och fick googla och följa en mängd olika guider för att få till det.

Allt jag använder för uppsättet finns dock under mitt git repo.

Eftersom att jag fick göra om många av stegen många gånger så använde jag mig av en mall som jag skrev under processens gång med inspiration från vfarcic(se referenser), som jag även lagt till i mitt git-repo, den ger en översikt över vad som skall till för att sätta upp miljön.

https://raw.githubusercontent.com/josoder/devops_exam/master/commands

Referenser:

<https://www.vip-consult.solutions/post/easy-docker-swarm-jenkins-continuous-deployment-at-scale>

<https://technologyconversations.com/2017/08/03/jenkins-master-as-a-docker-service-running-inside-a-docker-for-aws-cluster/>

<https://github.com/vfarcic/docker-flow-stacks>