QA-Deployment with K8S

How to deploy multiple QA environments with the help of K8S

Jan Baer

27. Juli 2020

What we had before

- 3 independent QA environments
- 3 cloned Bamboo plans for deploymnent
- 3 configuration files with fixed host-urls
- Docker-compose with 68 containers running inside on each host
- Very difficult to add more QA environments
- Hard to investigate when a feature was not deployed correctly

QA Deployment

QA1 VERBU-6224

Build #1283
Progress: 100%

Time: 3 Minutes and 2 seconds
Status: **Successful**

Assigned to [redacted]
Assigned today at 09:28

UN-ASSIGN

VERBU-JIRA number

DEPLOY TO QA1

QA2 VERBU-6051

Build #1203
Progress: 100%

Time: 2 Minutes and 30 seconds
Status: **Successful**

Environment free

ASSIGN TO ME

VERBU-JIRA number

DEPLOY TO QA2

QA3 VERBU-6166

Build #1254
Progress: 100%

Time: 2 Minutes and 18 seconds
Status: **Successful**

Environment free

ASSIGN TO ME

VERBU-JIRA number

DEPLOY TO QA3

What we wanted to reach

- At least 6 parallel QA environments
- Easier scalable
- Only one config for QA
- Better management and error investigation

QA Deployment

VERBU-JIRA number

CREATE NEW DEPLOYMENT

Feature: VERBU-6051 (active) **Created on:** 20.07.2020 15:23 (vor einem Tag) **Created by:** [redacted]

UPDATE DEPLOYMENT | REMOVE DEPLOYMENT

BU-DESKTOP | CUSTOMER AREA | GF-DESKTOP | MOBILE | GF MOBILE | CUSTOMER AREA | BUBOT | BU-TARIFF ADMIN | GF-TARIFF ADMIN | TEMPLE | WALLET

SHOW DETAILS

Feature: VERBU-6264 (active) **Created on:** 21.07.2020 11:53 (vor 8 Stunden) **Created by:** [redacted]

UPDATE DEPLOYMENT | REMOVE DEPLOYMENT

BU-DESKTOP | CUSTOMER AREA | GF-DESKTOP | MOBILE | GF MOBILE | CUSTOMER AREA | BUBOT | BU-TARIFF ADMIN | GF-TARIFF ADMIN | TEMPLE | WALLET

SHOW DETAILS

Feature: VERBU-6119 (active) **Created on:** 21.07.2020 14:31 (vor 5 Stunden) **Created by:** [redacted]

UPDATE DEPLOYMENT | REMOVE DEPLOYMENT

BU-DESKTOP | CUSTOMER AREA | GF-DESKTOP | MOBILE | GF MOBILE | CUSTOMER AREA | BUBOT | BU-TARIFF ADMIN | GF-TARIFF ADMIN | TEMPLE | WALLET

HIDE DETAILS

communicator	nexus.intern.bu.check24.de:5000/docker/communicator:VERBU-6119_latest	restartCount: 0	status: active
desktop-api	nexus.intern.bu.check24.de:5000/docker/desktop-api:VERBU-6119_latest	restartCount: 0	status: active
desktop-app	nexus.intern.bu.check24.de:5000/docker/desktop-app:VERBU-6119_latest	restartCount: 0	status: active
wallet	nexus.intern.bu.check24.de:5000/docker/wallet:master_latest	restartCount: 0	status: active
u-tipper	nexus.intern.bu.check24.de:5000/docker/u-tipper:master_latest	restartCount: 0	status: active
tofu	nexus.intern.bu.check24.de:5000/docker/tofu:master_latest	restartCount: 0	status: active

What is K3s?

K3s is a fully compliant Kubernetes distribution with the following enhancements:

- Packaged as a single binary. (less than 100 MB.)
- Lightweight storage backend based on sqlite3 as the default storage mechanism. etcd3, MySQL, Postgres also still available.
- Wrapped in simple launcher that handles a lot of the complexity of TLS and options.
- Secure by default with reasonable defaults for lightweight environments.

What is K3s?

- Simple but powerful “batteries-included” features have been added, such as: a local storage provider, a service load balancer, a Helm controller, and the Traefik ingress controller.
- Operation of all Kubernetes control plane components is encapsulated in a single binary and process. This allows K3s to automate and manage complex cluster operations like distributing certificates.
- External dependencies have been minimized (just a modern kernel and cgroup mounts needed).

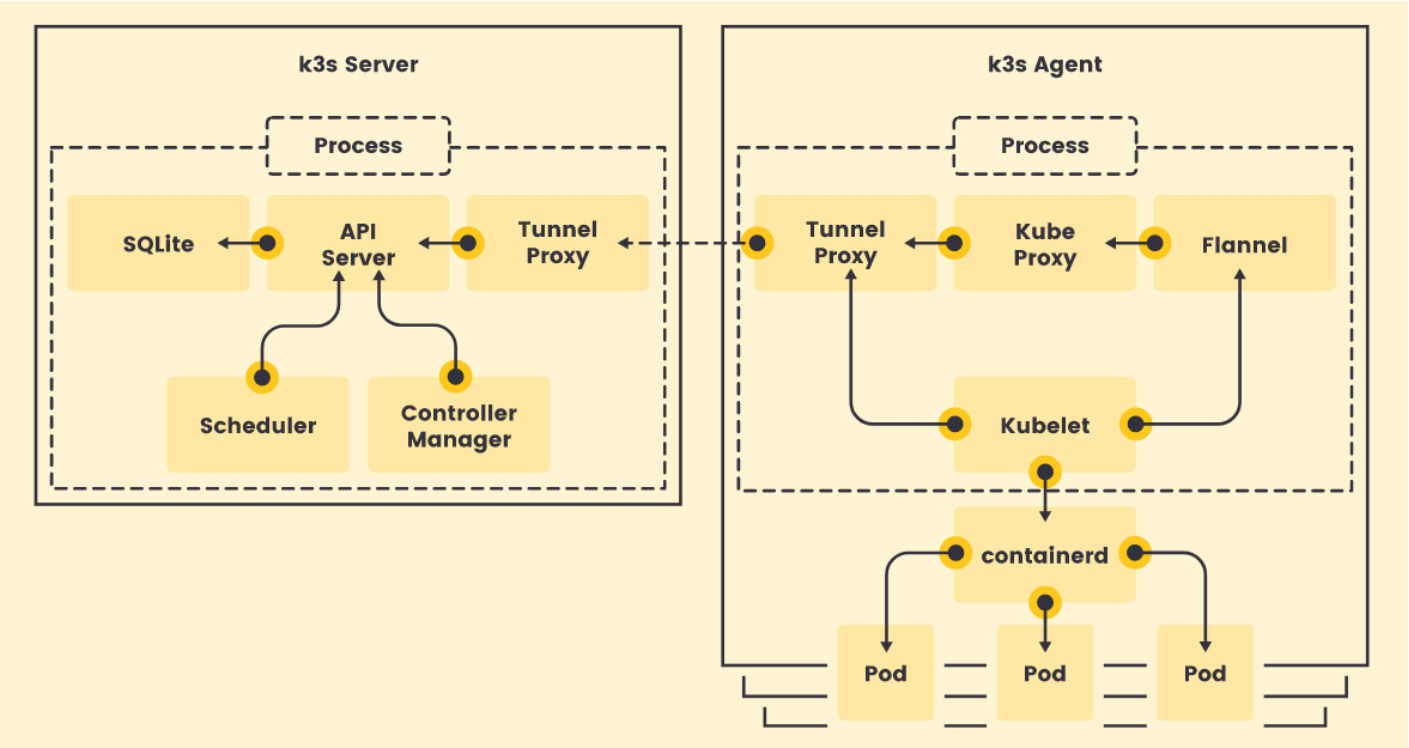
What's included in K3s

- Containerd
- Flannel
- CoreDNS
- CNI
- Host utilities (iptables, socat, etc)
- Ingress controller (traefik)
- Embedded service loadbalancer
- Embedded network policy controller

How to use K3s

- Uses per default Containerd as container-engine
- Can use alternatively Docker, but it's not required
- Run's as a Server and a Node on the same machine
- But also as Server(s) and Node(s) on separate machines
- You need at least one Server and one Node
- For high availability K3s supports a cluster of multiple servers

The architecture of K3s



Install K3s is very easy

Install the server

```
K3S_DATA_DIR=/data/k3s

export K3S_KUBECONFIG_MODE=644
export K3S_TOKEN="qa-k3s-cluster-1"

export INSTALL_K3S_EXEC="server --docker --data-dir ${K3S_DATA_DIR}"

curl -sfL https://get.k3s.io | sh -
```

Install the agent

```
export K3S_TOKEN="qa-k3s-cluster-1"

K3S_URL="https://[redacted]:6443"
K3S_DATA_DIR=/data/k3s

export INSTALL_K3S_EXEC="agent --server ${K3S_URL} --data-dir ${K3S_DATA_DIR} --docker"

curl -sfL https://get.k3s.io | sh
```

K3s will be installed as Systemd service

Server

```
● k3s.service - Lightweight Kubernetes
   Loaded: loaded (/etc/systemd/system/k3s.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-07-11 12:32:12 CEST; 2 weeks 2 days ago
     Docs: https://k3s.io
 Main PID: 1232 (k3s-server)
    Tasks: 0
   CGroup: /system.slice/k3s.service
           └─1232 /usr/local/bin/k3s server --docker --node-label agent-type=server --data-dir /data/k3s
```

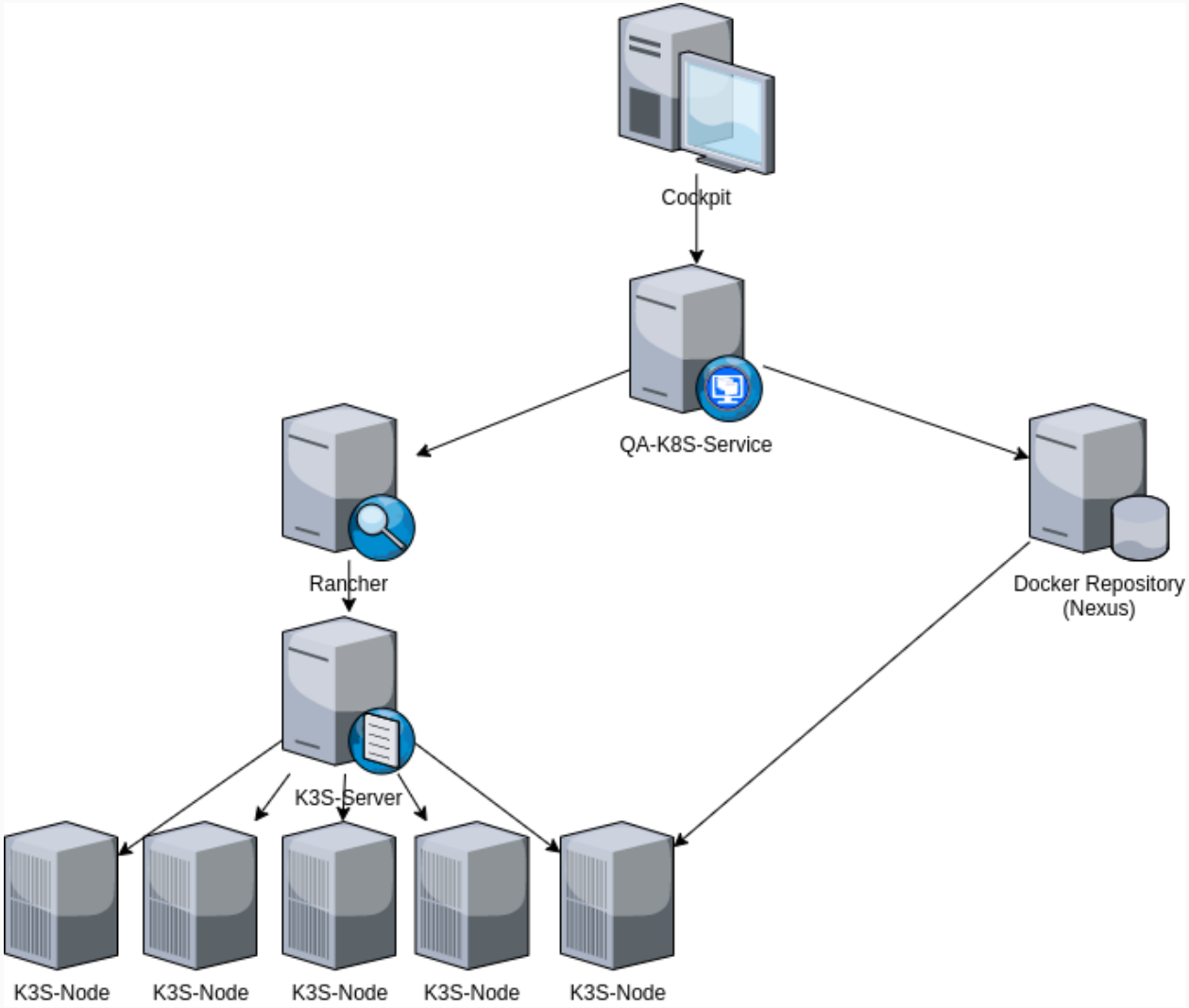
Agent

```
● k3s-agent.service - Lightweight Kubernetes
   Loaded: loaded (/etc/systemd/system/k3s-agent.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-07-24 15:50:56 CEST; 3 days ago
     Docs: https://k3s.io
 Main PID: 1328 (k3s-agent)
    Tasks: 0
   CGroup: /system.slice/k3s-agent.service
           └─1328 /usr/local/bin/k3s agent --docker --node-label agent-type=worker --data-dir /data/k3s
```

The whole K3s cluster

State	Name	Roles	Version	CPU	RAM	Pods
Active	bu-int-k8s-node-01 172.30.136.197 agent-type=worker	Worker	v1.18.3+k3s1 19.3.11	0.9/4 Cores	1.7/7.8 GiB	36/110
Active	bu-int-k8s-node-02 172.30.136.198 agent-type=worker	Worker	v1.18.3+k3s1 19.3.11	0.9/4 Cores	1.6/7.8 GiB	34/110
Active	bu-int-k8s-node-03 172.30.136.199 agent-type=worker	Worker	v1.18.3+k3s1 19.3.11	0.9/4 Cores	1.1/7.8 GiB	24/110
Active	bu-int-k8s-node-04 172.30.136.200 agent-type=worker	Worker	v1.18.3+k3s1 19.3.11	0.8/4 Cores	1.7/7.8 GiB	37/110
Active	bu-int-k8s-node-05 172.30.136.205 agent-type=worker	Worker	v1.18.3+k3s1 19.3.11	0.6/4 Cores	1/7.8 GiB	23/110
Active	bu-int-k8s-server-01 172.30.136.196 agent-type=server	Control Plane	v1.18.3+k3s1 19.3.11	0.1/2 Cores	0.1/3.9 GiB	7/110

How is it working together



What problems we had to solve

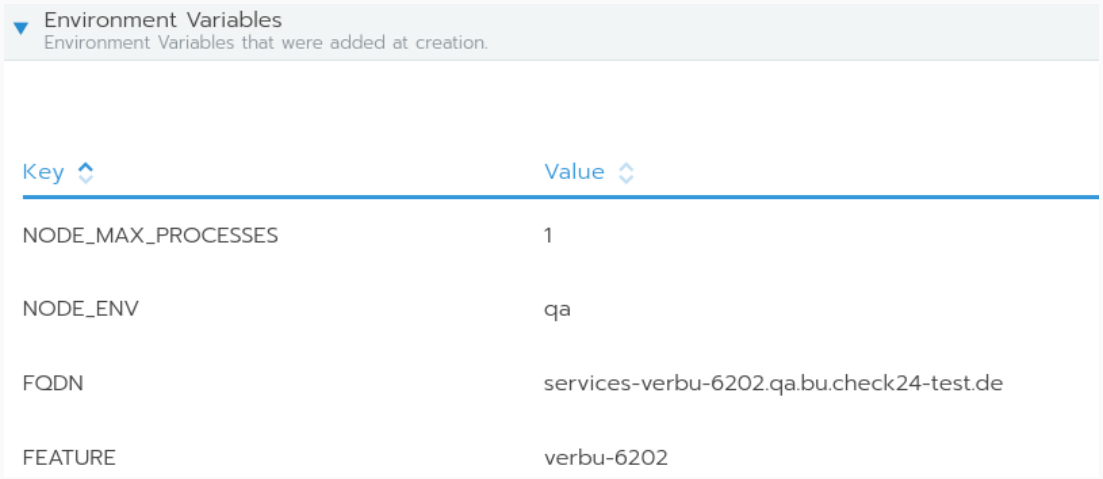
- Dynamic creation of urls
- Improve first-deployment and update and cleanup times
- Waiting for depending services (NSQ)
- Find the right limits
- Rewriting urls

Dynamic creation of urls

Use placeholders in config files

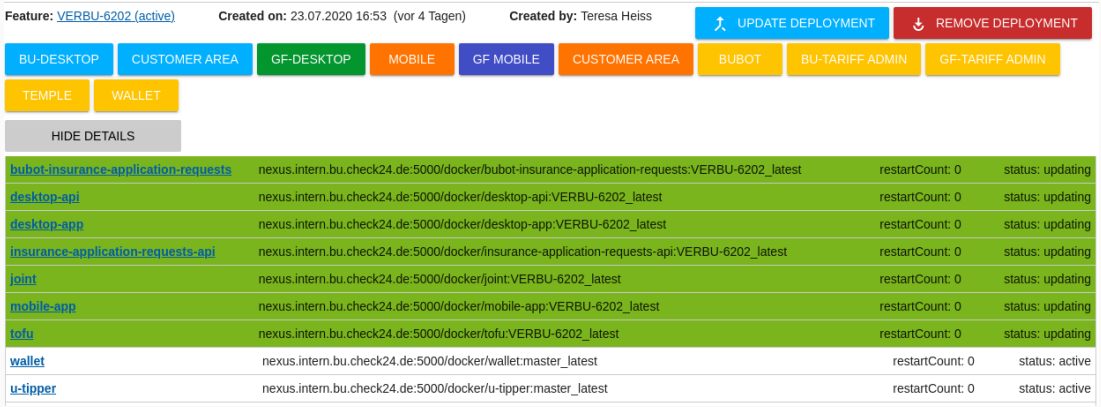
```
{
  "env": "qa",
  "origin": "https://services-${feature}.qa.bu.check24-test.de",
  "services": {
    "auth": "https://services-${feature}.qa.bu.check24-test.de/auth",
    "ripple": "https://services-${feature}.qa.bu.check24-test.de/ripple",
    "miami": "https://services-${feature}.qa.bu.check24-test.de/miami",
    "customerActivities": "https://services-${feature}.qa.bu.check24-test
  },
```

Set feature as environment variable



Improve deployment times

- First deployment takes a while because it requires to deploy ~70 Pods
- Only update what has changed



- No graceful shutdown reduces deletion time

```
terminationGracePeriodSeconds: 0
```

Waiting for dependent services

- Some of the services requiring a running NSQ service

```
<% if ( requiresNsq ) { %>
initContainers:
  - name: wait-for-nsq
    image: subfuzion/netcat
    command: ['sh', '-c', "while ! nc -z nsqd 4151; do sleep 0.5; done"]
<% } %>
```

State	Name	Image	Restarts
Waiting PodInitializing	brain	nexus.intern.bu.check24.de:5000/docker/brain:master_latest	0
Waiting PodInitializing	wait-for-nsq Init Container	subfuzion/netcat	

Finding the right limits

- Observe a deployment to know what resources are required

```
watch -n 2 -t kubectl top pods -n verbu-6202
```

NAME	CPU(cores)	MEMORY(bytes)
accounting-api-5588fc9678-pd57h	2m	65Mi
acid-6c84c7dff-cdm52	1m	63Mi
addressservice-5b5bf7f4c7-4npfw	1m	97Mi
auth-cb4c45b5b-522f5	9m	62Mi
auth-ui-6cccc7b664-kwgq5	1m	80Mi
brain-5c85f89b47-l6hlw	2m	64Mi
bu-cleanup-fd45f8b85-4×75f	1m	68Mi
bubot-748c6b76cb-tpwqx	12m	110Mi
bubot-accounting-55bf8c85cf-7fc9w	1m	70Mi
bubot-appointments-6796479ccb-khst9	1m	93Mi
bubot-consulting-process-6646dd9665-t9fzp	1m	88Mi
bubot-documents-5fcd6f856-ksc5g	24m	105Mi
bubot-insurance-application-requests-66b787796b-97rzd	1m	96Mi
bubot-mailing-5666f7dcc5-vvmxk	1m	88Mi
bubot-rivo-586dd8f69b-zgxx7	1m	71Mi
bubot-salary-6cf7f5d85f-lh929	1m	67Mi
bus-5b86d84798-8xpfg	1m	49Mi
coachman-94d74db9c-vd4sz	22m	76Mi
communicator-7cd7d4968b-zk7rp	1m	74Mi
consulting-process-api-549fcb8986-r8zzb	5m	66Mi

Some urls has to be changed

- Remove `/eventbus` from the url before forwarding to NSQ service

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/ingress.class: traefik
    traefik.frontend.rule.type: PathPrefixStrip
  name: ingress-eventbus
  namespace: <%= namespace %>
spec:
  rules:
  - host: <%= host %>
    http:
      paths:
      - backend:
          serviceName: <%= name %>
          servicePort: <%= port %>
        path: <%= path %>
  tls:
    - hosts:
      - <%= host %>
```

Next steps

- One MongoDb per feature deployment
- Improve visualisation of the deployment state

Which problems we still have

- Too many pods for every feature deployment
- Deployment becomes unstable after the 6th deployment and it's unclear why

Questions?

Thank you!