Distributed applications: A challenge for systems, networks, and application development

Holger Karl
Paderborn University
Claims

• Developing and deploying applications: Together
  • Buzzword: Continuous software engineering, CI/CD

• Deploying: Applications and infrastructure
  • Buzzword: Infrastructure as code

• Education issues
Key ingredients

Applications

Distributed Cloud Computing

Software-defined networking

Network Function Virtualization

CleanSky 2017, Göttingen
Developing old school

- Monolithic IDEs producing monolithic code
Deploying?

Code

???

CleanSky 2017, Göttingen
Deploying, old school

- Great for single application, single machine

```bash
$ # download
$ wget http://www.developer.com/app.tgz
$ tar xzf app.tgz
$ # compile and build
$ cd app
$ ~/app> ./configure
$ ~/app> make
$ ~/app> make install
$ # run
$ ~/app> ./app
```
Developing, ca. 2017

- Own code, plus tons of modules, libraries, **microservices**, ...
Deploying using virtual machines

- Great for simple applications
- Even inside, e.g., VMWare clusters

Customize a ready-made image to local needs
E.g., network addresses

```bash
$ # download description
$ wget http://www.developer.com/app.tgz
$ tar xzf app.tgz
$ cd app
$ # edit configuration:
$ vi ansible-playbook.yml
$ # download actual image, configure it
$ vagrant --provision up
$ # tell DNS where to find the app:
$ curl --request POST www.mydns.com -d {'fqdn': myapp.com, 'ip': 1.2.3.4}
```
Setup after deployment
Deploying complex applications – Manually

- Anything real will need multiple machines

```
$ wget http://www.developer.com/app.tgz
$ tar xfz app.tgz
$ cd app/frontend
$ # complex editing of network configuration
$ app/frontend> vi ansible-playbook.yml
$ app/frontend> vagrant --provision up

$ cd ../app/backend
$ # complex editing of network configuration
$ app/backend> vi ansible-playbook.yml
$ app/backend> vagrant --provision up
$ # tell DNS where to find the app:
$ curl --request POST www.mydns.com -d {'fdqm': 'myapp.com', 'ip': frontend_ip}
```

- Getting slightly less great
Setup after deployment

- DNS
- Front-end VM
- Back-end VM

CleanSky 2017, Göttingen
Scaling out?

- Load increases: need another frontend

Start second frontend

Integrate it with backend

```
$ # copy frontend configuration:
$ cp -r app/frontend app/frontend2
$ cd app/frontend2
$ app/frontend2> # complex editing of network configuration
$ app/frontend> vi ansible-playbook.yml
$ # bring up a new frontend instance
$ app/frontend> vagrant --provision up
$
$ # tell backend to accept new frontend
$ cd ../app/backend
$ app/backend> vagrant ssh [backend] $ vi backend.config
$ [backend] $./backend restart

# add the second frontend
# trigger reloading of config

CleanSky 2017, Göttingen
12
Scaling out?

• But wait, there’s more:

```
$ wget http://www.loadbalancer.com/lb.tgz
$ tar xzf lb.tgz
$ cd lb
$ # complex editing of lb configuration
$ lb> vagrant --provision up
$ cd ..
$ ssh firewall
$ # tell DNS where to find the app:
$ curl --request POST www.mydns.com -d {'fdqn': 'myapp.com', 'ip': lb_ip}
```

- Firewall needs to be told to allow traffic
- DNS should point to loadbalancer
- Loadbalancer needed
Setup after deployment

VM

Load-balancer

DNS

VM

Front-end1

Back end

VM

Front-end2

CleanSky 2017, Göttingen
Frontends at different sites?

• Wouldn't this make more sense?
Deploying frontends at remote sites

• Definitely not great!

$ # configure frontend2 to run elsewhere
$ cd app/frontend2
$ vagrant halt
$ # put IP of other site into configuration
$ vagrant up
$ # drop loadbalancer
$ # reconfigure firewall at both sites
$ # reconfigure DNS
$ # reconfigure backend

CleanSky 2017, Göttingen
What do we need?

• Get rid of manual configuration via console

• **Describing** configuration of a composed application
  • Its actual application-oriented components
  • Support components like firewalls, …
  • Scaling properties

• **Deploying** such a composed application

• **Configuring** underlying infrastructure
Issues

Describe \(\rightarrow\) Deploy \(\rightarrow\) Configure

CleanSky 2017, Göttingen
Describing complex applications – Templates

- **Templates** describe components in an application
- Needs
  - Which components
  - How interconnected
  - How to scale
  - Where to place
Template examples

• OpenStack: HEAT Orchestration Template (HOT)
  • HEAT: OpenStack’s orchestration plugin
  • Composition, interconnected, limited scaling properties, single site

• Docker Swarm
  • Container-based (instead of VMs)
  • Relatively simple setup, easy generation

• Google Kubernetes
Docker/Kubernetes example: Guestbook application

https://github.com/kubernetes/kubernetes/blob/release-1.2/examples/guestbook/all-in-one/guestbook-all-in-one.yaml
Templates, as they should be

- **Quantitative annotations**
  - From profiling
- Scaling options
- …

CleanSky 2017, Göttingen
Services?

• “Ordinary” application boxes?

• “Network” functions?

• What’s the key, pivotal difference?
Claim: None!

- There is no inherent difference between
  - microservices
  - network functions

- Differences are a historical, ecosystem-based coincidence
  - E.g., separate orchestrators, …
Mid-term goal: Merge NFV and (D)CC

DCC

NFV
Crossing from VNFs to SDN?

- Example: Load balancer
  - Given as a VNF
  - Or as SDN rules

- Decide which one?
- Automatically convert?
- Open!
Mid-term goal: Merge NFV, DCC, SDN
Applications become more complex

- **Application** = Actual application + networking

![Diagram](image)

- Application components/microservices
- Networking components (SDN or NFV)

CleanSky 2017, Göttingen
Remember

Developer knows best!
Example template: Quantitative scaling

M. Keller, C. Robberts, H. Karl: Template Embedding: Using Application Architecture to Allocate Resources in Distributed Clouds, UCC 2014
Example template: Reordering

S. Mehraghdam, M. Keller, H. Karl: Specifying and Placing Chains of Virtual Network Functions, CloudNet 2014
Describe $\rightarrow$ Deploy $\rightarrow$ Configure

Issues

1000
Firewall $\rightarrow$ Load-balancer
200
Front-end
5000
Back-end $\rightarrow$ Database

CleanSky 2017, Göttingen
Deploying?

• Scaling?
• Placement?
• Lifecycle management?
  • When to start, stop, migrate state of a component

• Where to put which component?
• Virtual network embedding – NP-hard
• Embedding scalable templates: generalization
  • Still solvable with good heuristics

CleanSky 2017, Göttingen
One-size-fits-all solution?

- Use a single, “perfect” deployment scheme?
  - Independent of application?

- Remember: Developer knows best!
  - SotA: Some parameterization exists, but limited in flexibility
    - E.g., Kubernetes

- No! Need bespoke deployment, orchestration schemes!
Developer knows best!

- **Application = Orchestration + actual application + networking**
  - UNIFY, SONATA, ...

![Diagram showing the components of an application: Front-end, Backend, Load-balancer, Firewall, Place-ment, Life-cycle. The diagram is labeled with 'Bespoke orchestration!' and shows the relationship between these components.]

CleanSky 2017, Göttingen
Reality check: Networking

- Networking: SDN controller applications
- Create network topology for application
- Developer support? Legacy?

Bespoke networking!
High-level architecture

SDN controller

Application
Front-end
Firewall
Load-balancer
Back-end

Orchestrator
NFV orchestrator
Placement
Life-cycle
Cloud orchestrator

CleanSky 2017, Göttingen
Reality check: Developer

Developer knows best?
Developer knows best?

Application

Front-end

Back-end

Load-balancer

Place-ment

Life-cycle

CleanSky 2017, Göttingen
Reality check: Develop SDN network applications

- NetIDE: Bring SDN closer to developers
- Supporting legacy applications on controller frameworks

NetIDE: All-in-one framework for next generation, composed SDN applications, NetSoft Demo, 2016

http://www.netide.eu
NetIDE: Ease developer’s tasks by SDK

Reality check: Templates

- Ideal developer writes perfect templates
- Real developer?

- Option: semi-automatic generation?
  - Based on semantic understanding of components

SFB 901
ON-THE-FLY COMPUTING

https://sfb901.uni-paderborn.de
Reality check: Orchestration with 3rd party modules?

• Run code from outside application inside an orchestration platform?
  • Security, isolation, conflicts, …?

• Sure – as microservices!
  • SONATA (5G PPP): microservices-based orchestration

http://sonata-nfv.eu
SONATA: Architecture overview

SONATA Service Platform

MANO Framework

NFVO
- Executive Message Broker
- Service Decision
- Service Execution
- Conflict resolution
- Resource Management
- Slice Management

Infrastructure Abstraction

Message Broker

VNF Adaptor

Monitoring processor plugin

Repositories
- Monitoring
- SSM
- Service
- UE
- M2M

Platform Catalogues

Gatekeeper

VIM

Infrastructure

VNF

VNF

CleanSky 2017, Göttingen
Challenge: Education!

• Teaching curricula: Integrate networking, distributed systems, virtualization

• Continuous education
Conclusion

• NFV and (D)CC need to merge; SDN integrated

• Developers and Ops need to understand that
  • DevOps for complex, distributed applications

• Research & education task