Challenges in Infrastructure-as-Code

Efficiency, Decentralization and Formalization

by Hélène Coullon (IMT Atlantique, Inria, LS2N - Nantes, France)
on 2025-06-18, Lille
DisCoTec-wide Keynote

» Outline

» Past

- **Since 2016** Associate professor at IMT Atlantique
- **2020-2022** 20% Adjunct professor at the Arctic university of Norway, Tromsø
- **2016-2021** Inria research chair

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

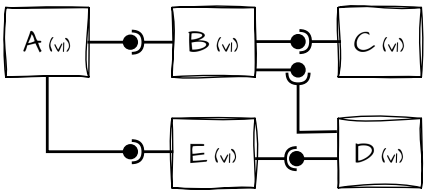
Inria
INVENTEURS DU MONDE NUMÉRIQUE

THE ARCTIC UNIVERSITY OF NORWAY • UIT

Infrastructure-as-Code

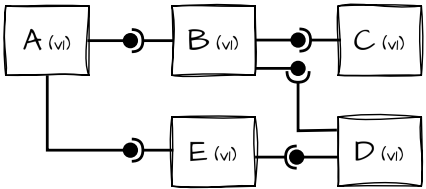
* Distributed software systems and their deployment
* Infrastructure-as-Code
* Coordination and declarativity in IaC
* Autonomic IaC

» Service-Oriented (SO) Distributed systems (DS)



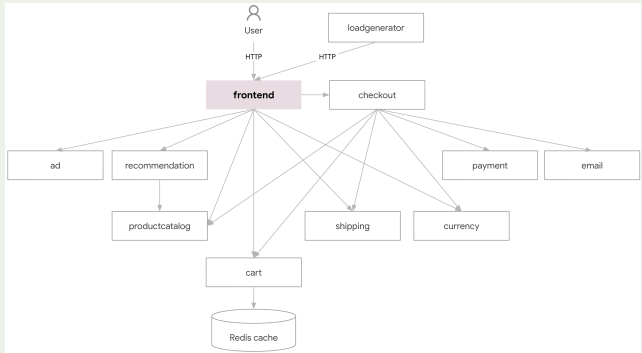
* Loose coupling components
* Use/provide interfaces for composition

» Service-Oriented (SO) Distributed systems (DS)

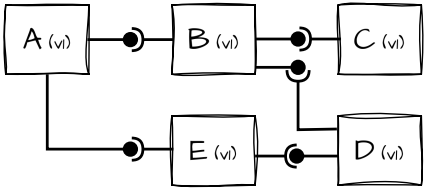


- Loose coupling components
- Use/provide interfaces for composition

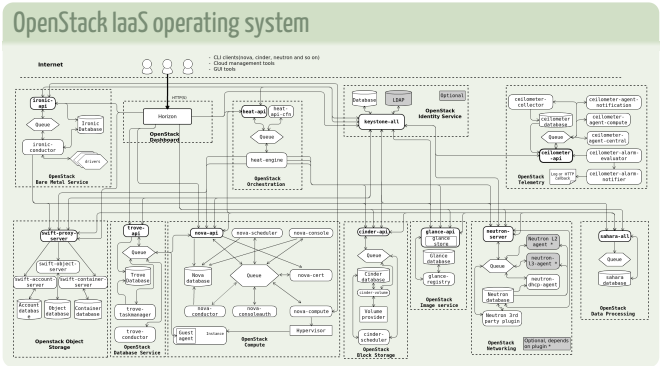
Microservices architecture - Online boutique



» Service-Oriented (SO) Distributed systems (DS)



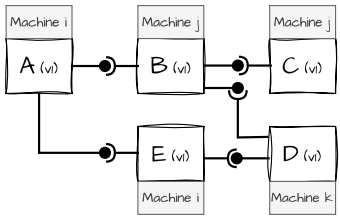
- Loose coupling components
- Use/provide interfaces for composition

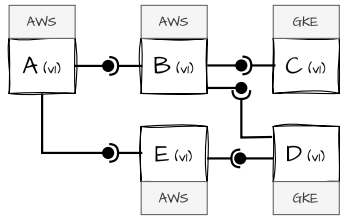


Designing and writing such SO systems: CBSE, SOA, micro-services architectures, etc.

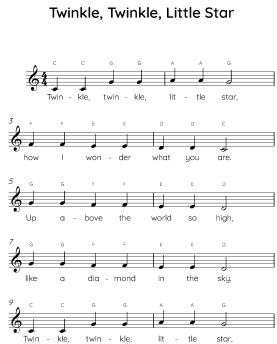
» Deploy and operate SODS

Distributed systems live on distant machines or platforms

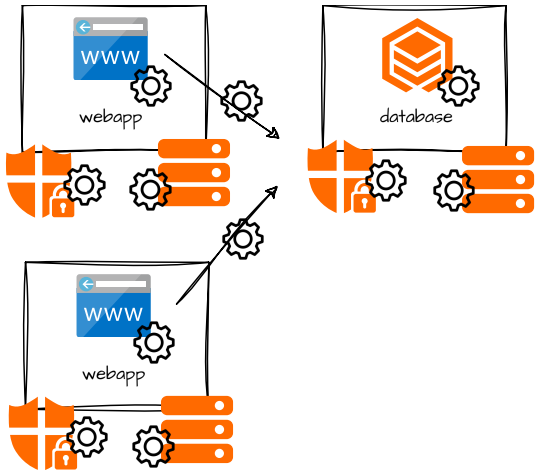




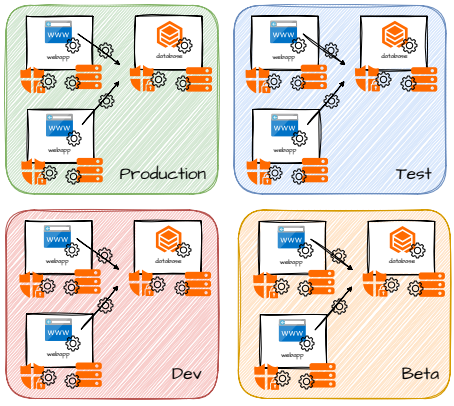
» Why is it complex?



» Why is it complex?

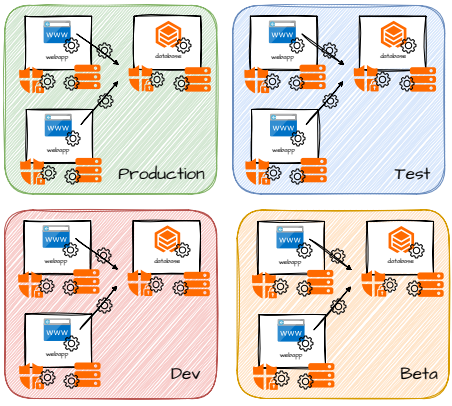


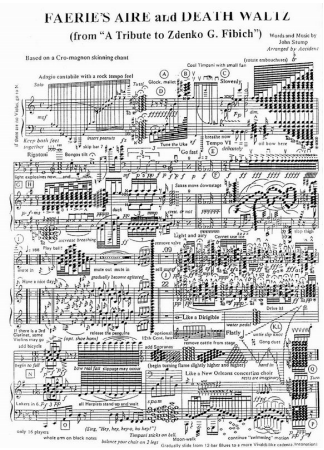
» Why is it complex?



+ operate

» Why is it complex?





+ operate

» Infrastructure-as-Code

Infrastructure

Infrastructure refers to the software, platform, or hardware that delivers or deploys applications [3]

» Infrastructure-as-Code

Infrastructure

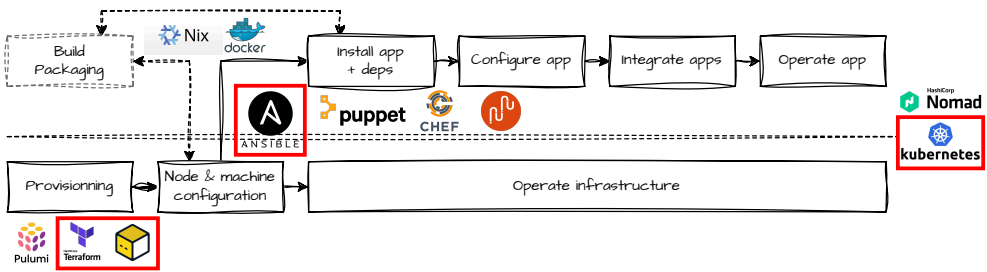
Infrastructure refers to the software, platform, or hardware that delivers or deploys applications [3]

Infrastructure-as-Code

See infrastructure deployment and management as programs or codes

- programming languages (DSL)
- versioned infrastructures
- testable infrastructures
- shareable infrastructures

» Infrastructure-as-Code in practice



» Ansible

main.yml

```yaml
- name: Common setup
  hosts: all
  roles:
    - common
- name: Install database server
  hosts: db
  roles:
    - database
```

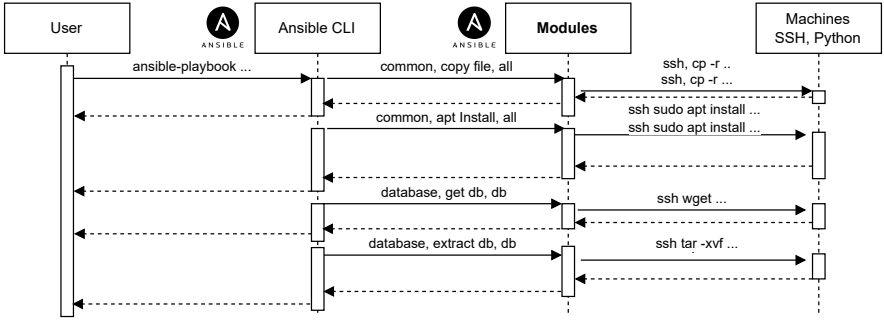
roles/common/tasks/main.yml

```yaml
- name: common
  tasks:
    - name: copy file
      ansible.builtin.copy:
        src: /tmp/file.txt
        dest: /tmp/file.txt
    - name: apt Install
      ansible.builtin.apt:
        name: "python3"
        state: present
```

roles/database/tasks/main.yml

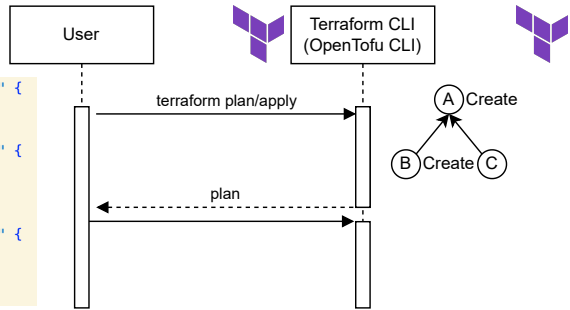
```yaml
- name: database
  tasks:
    - name: get db
      ansible.builtin.get_url:
        url: "https://example.com/database.tar.gz"
        dest: "/tmp/database.tar.gz"
    - name: extract db
      ansible.builtin.unarchive:
        src: "/tmp/database.tar.gz"
        dest: "/opt/database"
        remote_src: yes
```

» Ansible

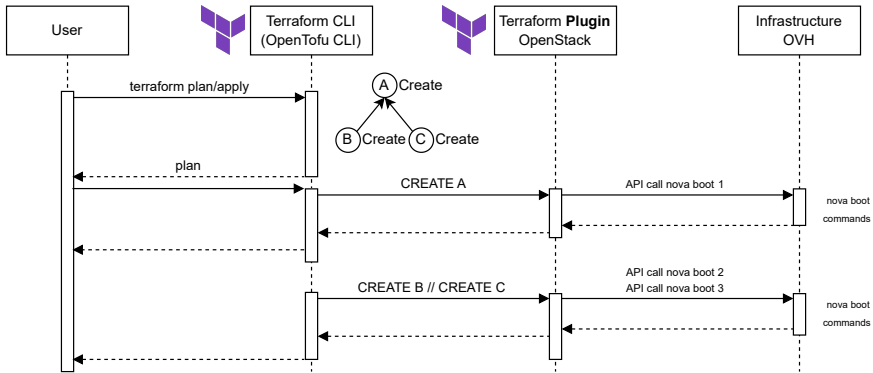

User
ANSIBLE
Ansible CLI
ANSIBLE
Modules
Machines
SSH, Python
ansible-playbook ...
common, copy file, all
ssh, cp -r ..
ssh, cp -r ...
common, apt Install, all
ssh sudo apt install ...
ssh sudo apt install ...
database, get db, db
ssh wget ...
database, extract db, db
ssh tar -xvf ...

» Terraform/OpenTofu

```
1  resource "openstack_compute_instance_v2" "A" {
2    # attributes for instance A
3  }
4  resource "openstack_compute_instance_v2" "B" {
5    # attributes for instance B
6    depends_on = [A]
7  }
8  resource "openstack_compute_instance_v2" "C" {
9    # attributes for instance C
10   depends_on = [A]
11 }
```

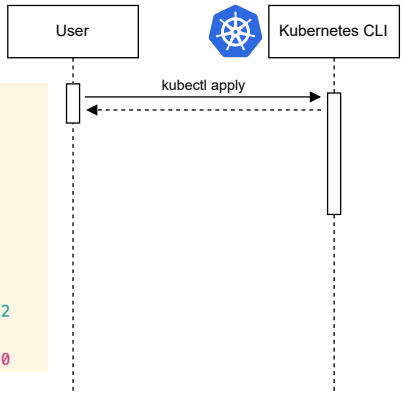


» Terraform/OpenTofu

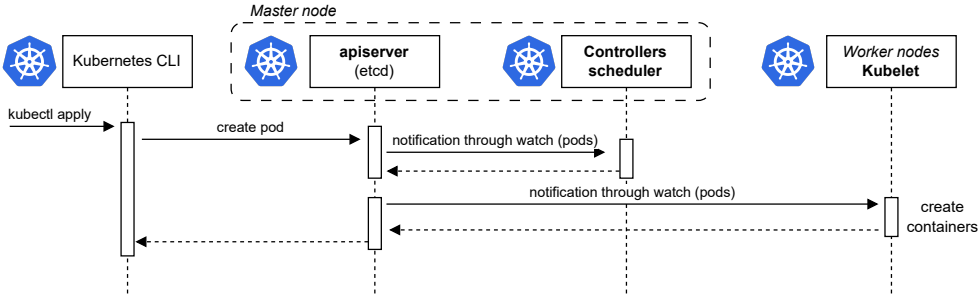


» Kubernetes

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
    role: webserver
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```



» Kubernetes



"Kubernetes in action", Marko Lukša[1]

[1] https://sutlib2.sut.ac.th/sut_contents/H173702.pdf

» Comparison

	Coordination	Actions	Actuators
Ansible	sequence+spmd	modules	modules + ssh
Terraform	DAG of actions	CRUD	plugins + APIs
Kubernetes	embarassingly	CRUD	controllers + apiserver + kubelet

» Declarative IaC

Terraform and Kubernetes are declarative languages!

Specify what is wanted, not how to get it!

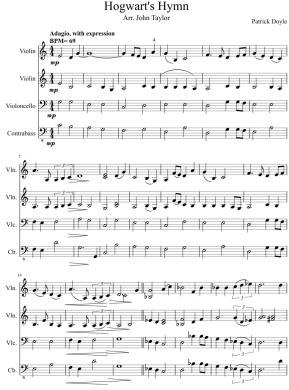
» Declarative IaC

Terraform and Kubernetes are declarative languages!

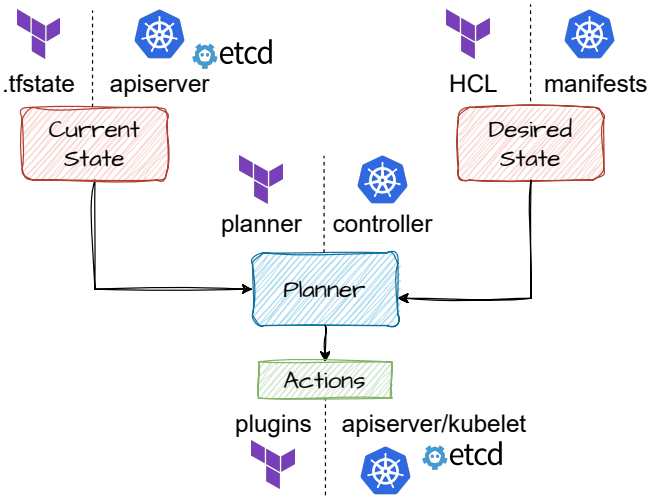
Specify what is wanted, not how to get it!

I want music that sounds magical

→



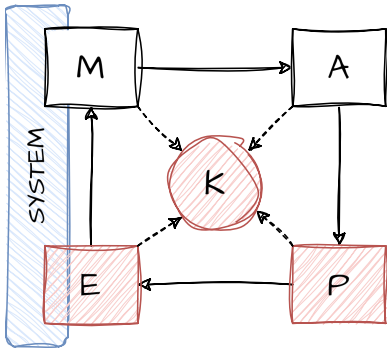
» Declarativity: a matter of state and plan



» Comparison

	Coordination	Actions	Actuators	Declarative
Ansible	sequence	modules	modules + ssh	✗
Terraform	DAG	CRUD	plugins + APIs	✓
Kubernetes	embarassingly	CRUD	controllers + apiserver	✓

» Autonomic IaC (reconfiguration)



- M = Monitor
- A = Analyze
- P = Plan (Declarativity)
- E = Execute
- K = Knowledge

Orchestration tools achieve the full reconciliation loop for specific cases like auto-scaling, or faults.

"The vision of autonomic computing" Kephart et. al., 2003[1]

» Comparison

	Coord.	Actions	Actuators	Declarative	mapek
Ansible	sequence	modules	mod. + ssh	✗	E
Terraform	DAG	CRUD	plugins	✓	PEK
Kubernetes	embar.	CRUD	controllers	✓	MAPEK

Efficiency

* Why efficiency?
* Concerto
* How to make Concerto declarative?

» Why efficiency?

1. Mitigate or recover quickly from a critical situation
   * faults, unavailabilities
   * security issues

» Why efficiency?

1. Mitigate or recover quickly from a critical situation
   * faults, unavailabilities
   * security issues
2. Quickly reach the new desired state
   * performance, QoS, QoE

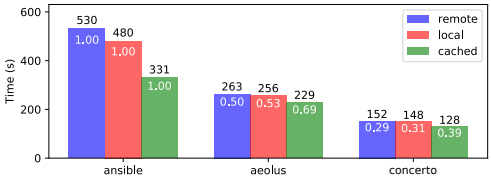
» Why efficiency?

1. Mitigate or recover quickly from a critical situation
   * faults, unavailabilities
   * security issues
2. Quickly reach the new desired state
   * performance, QoS, QoE
3. Accelerate the deployments and change of large systems

» Why efficiency?

Minimal OpenStack deployment

* 11 services



Artifacts Grid'5000

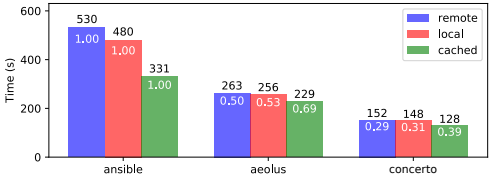
Up to 70% gain compared to Ansible

During release periods OpenStack can be deployed hundreds of times a day

» Why efficiency?

Minimal OpenStack deployment

* 11 services



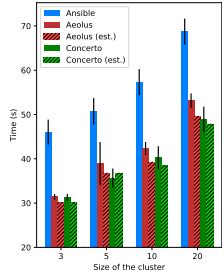
Artifacts Grid'5000

Up to 70% gain compared to Ansible

During release periods OpenStack can be deployed hundreds of times a day

Reconf. multi-site OpenStack

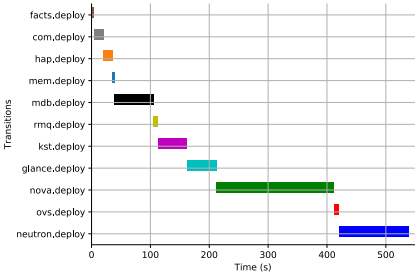
* OpenStack Summit 2018
* Galera cluster of DB

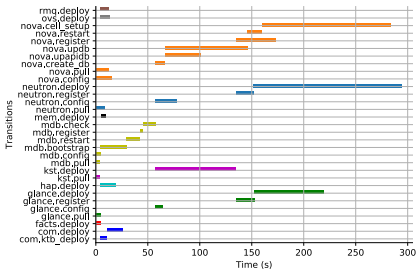


Artifacts Grid'5000

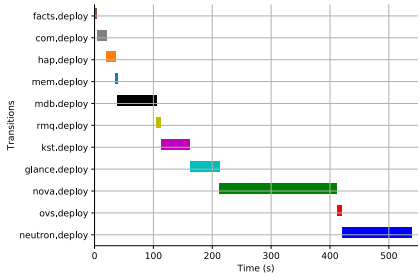
Up to 40% gain compared to Ansible

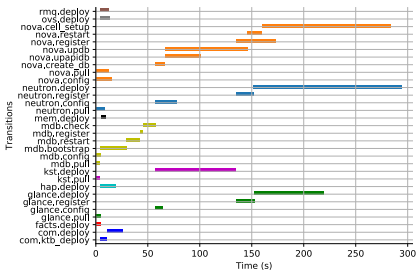
» Parallelism in reconfiguration





» Parallelism in reconfiguration





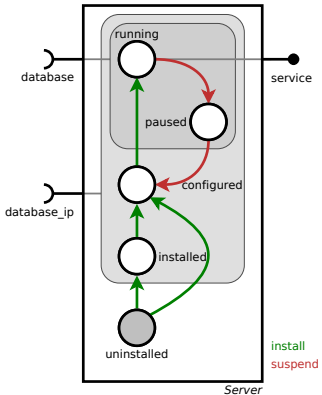
Reconfigurations are DAGs (Directed Acyclic Graphs) of (system) actions

However,

* Life cycle abstraction is required for developers, DevOps, and (P)
* Flexible enough abstractions to not lose opportunities for parallelism

» Concerto - Control components

Programmable life cycle with fine-grained dependencies



Written by the component developers

Internal net: Models the life cycle of *a* component

- places = milestones
- transitions = concrete *actions* to perform

Interfaces

- ports (CBSE)
  - use ports = requirements
  - provide ports = provisions
  - during execution: *active/inactive*
- behaviors (subset of transitions)
  - *actions* on the life cycle

» Concerto - Reconfiguration language

Interact with the components' life cycles

Used by DevOps, or a declarative tool

Create assemblies and make them evolve at runtime (CBSE)

- `add/delete` a component instance
- `connect/disconnect` two component instances

» Concerto - Reconfiguration language

Interact with the components' life cycles

Used by DevOps, or a declarative tool

Create assemblies and make them evolve at runtime (CBSE)

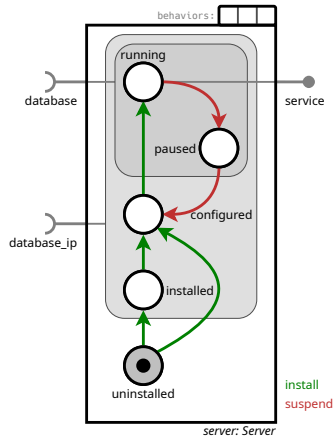
- `add/delete` a component instance
- `connect/disconnect` two component instances

Interact with the lifecycle of components

- `push` a behavior to the behavior queue on a component instance
- `wait` for a given component instance or wait for all components

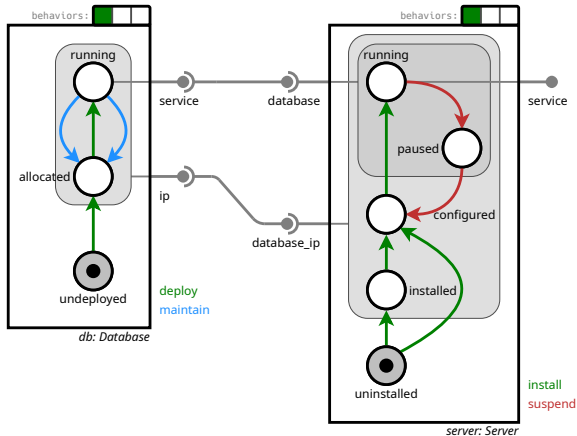
» Deployment example

```
add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)
```



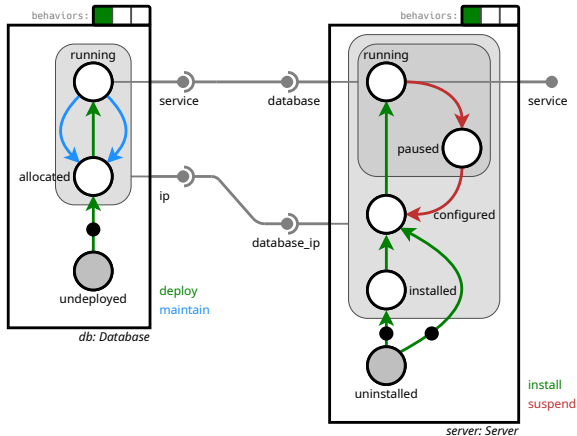
» Deployment example

add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)



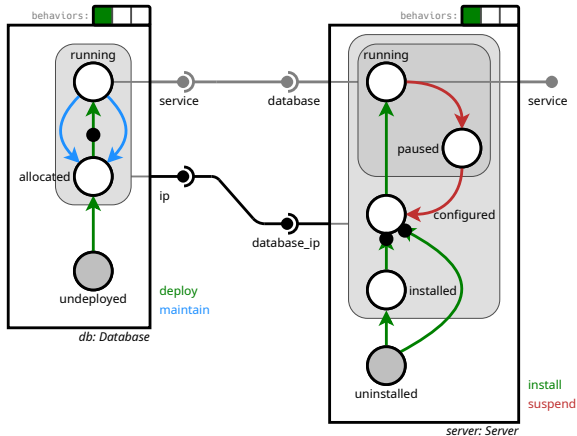
» Deployment example

```
add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)
```



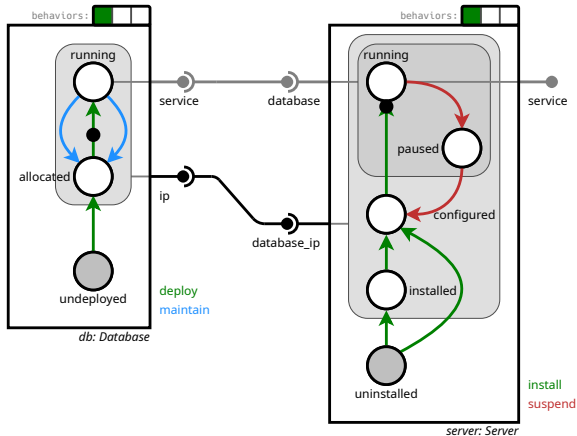
» Deployment example

add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)



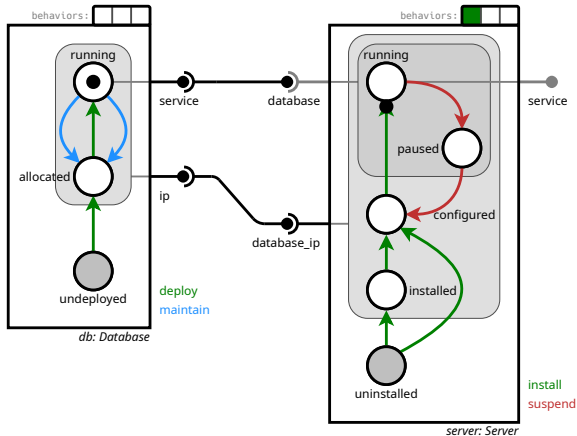
» Deployment example

add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)



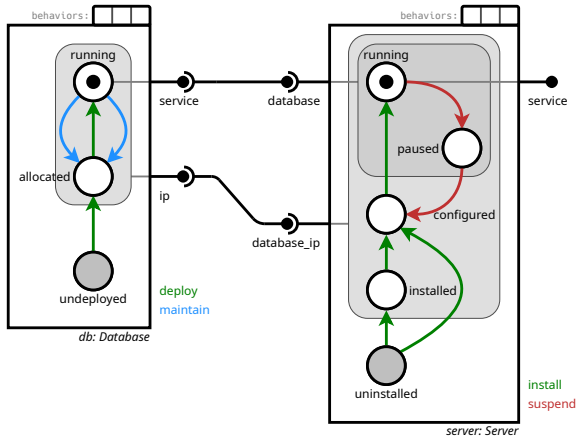
» Deployment example

add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)



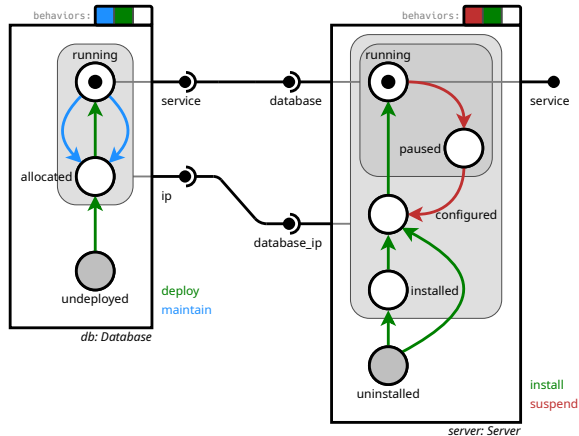
» Deployment example

```
add(server: Server)
add(db: Database)
con(server.database_ip,db.ip)
con(server.database,db.service)
pushB(server,install)
pushB(db,deploy)
wait(server)
```



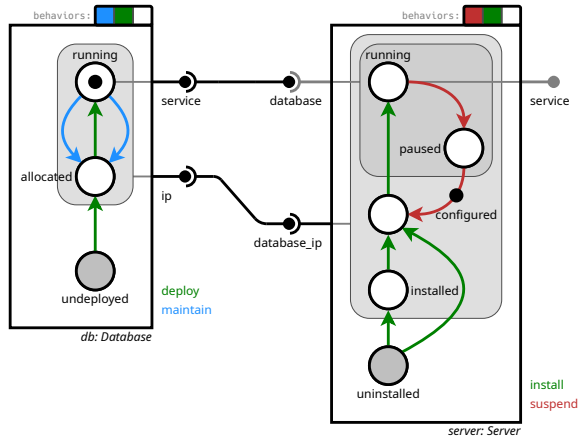
» Maintenance example

pushB(db,maintain)
pushB(db,deploy)
pushB(server,suspend)
pushB(server,install)
wait(server)



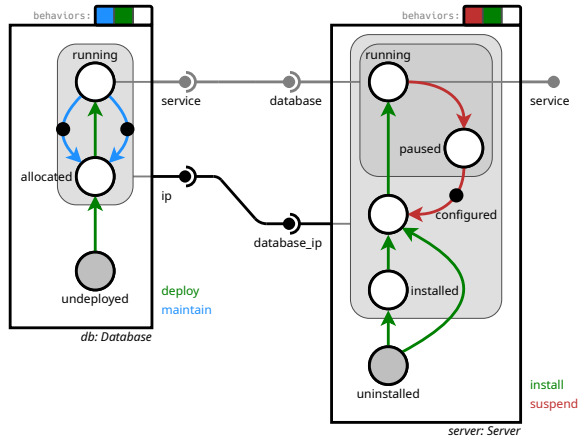
» Maintenance example

pushB(db,maintain)
pushB(db,deploy)
pushB(server,suspend)
pushB(server,install)
wait(server)



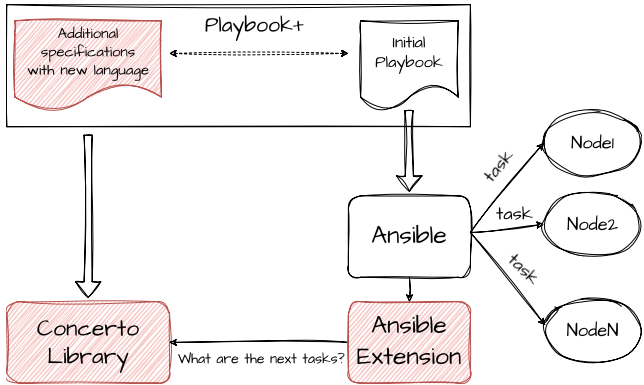
» Maintenance example

pushB(db,maintain)
pushB(db,deploy)
pushB(server,suspend)
pushB(server,install)
wait(server)



» Leveraging Concerto in Ansible

The CoAnsible project (INRIA transfer action)



» How to make Concerto declarative?

Principle

- **inputs:** current state of the system (M) + reconfiguration goals (A)
  - set of behaviors to execute on designated components
  - constraints on the final state of ports
- **output:** a Concerto reconfiguration program
  - `pushB` requests
  - `wait` commands
- About creations/deletions/(dis)connections
  - usual to handle topological changes before and after behavioral requests and synchronizations (e.g., deployment)

Problem formulation

Find a valid (optimal) schedule of `pushB` and `wait` instructions

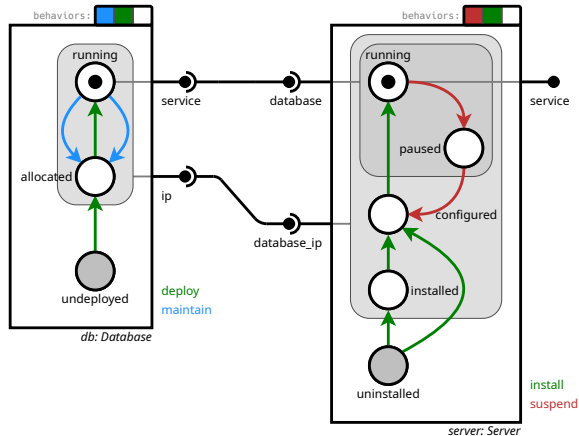
» Maintenance example

Goal

behaviors:
- ***component***: *db*
  behavior: *maintain*

components:
- ***forall***: *running*

pushB(db, maintain)
pushB(db, deploy)
pushB(server, suspend)
pushB(server, install)
wait(server)



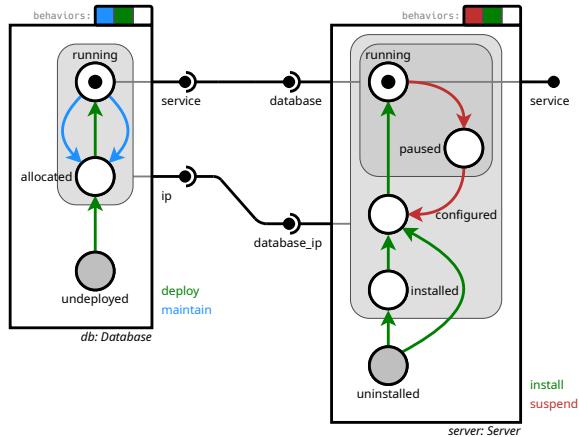
» Maintenance example

Goal

behaviors:
- ***component***: *db*
  behavior: *maintain*

components:
- ***forall***: *running*

pushB(db, maintain)
pushB(db, deploy)
pushB(server, suspend)
wait(db)
pushB(server, install)
wait(server)



» Leveraging Concerto in Terraform?

Offering programmable life cycles instead of fixed CRUD in Terraform

» Leveraging Concerto in Terraform?

Offering programmable life cycles instead of fixed CRUD in Terraform

* New plugin framework for programmable life cycles
* The planner of Terraform should be replaced with our planner
* The execution of Terraform (apply) should be able to use the Concerto library

» Leveraging Concerto in Terraform?

Offering programmable life cycles instead of fixed CRUD in Terraform

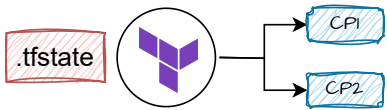
* New plugin framework for programmable life cycles
* The planner of Terraform should be replaced with our planner
* The execution of Terraform (apply) should be able to use the Concerto library

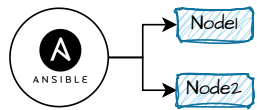
More difficult than CoAnsible?

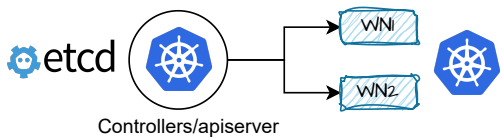
Decentralization

* Why decentralization?
* From Concerto to Ballet

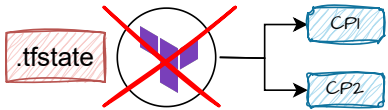
» Fault-tolerance of IaC tools

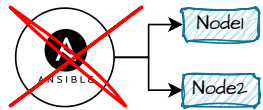


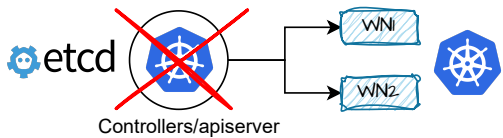




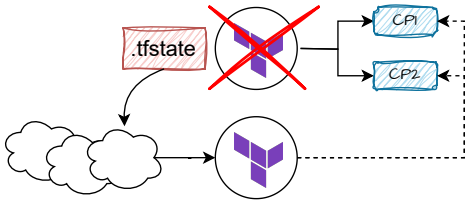
» Fault-tolerance of IaC tools

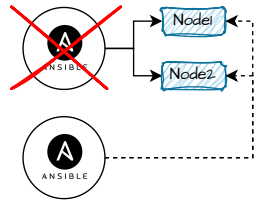


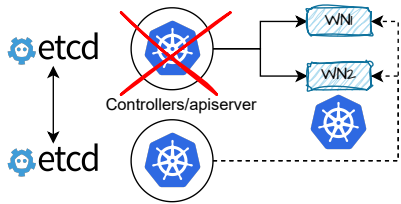




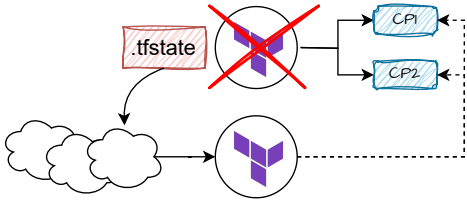
» Fault-tolerance of IaC tools

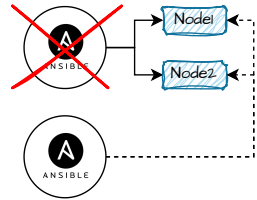


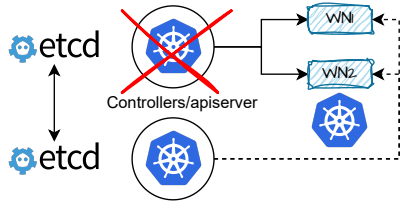




» Fault-tolerance of IaC tools







⚠ idempotence is required $f(f(x)) = f(x)$

» Resilience of IaC tools

Only a matter of replication?

I think the answer is “no”

» Resilience of IaC tools

Only a matter of replication?

1. At the Edge or in constrained CPS, disconnection is the norm
   * any distant central node is regularly unreachable (but still alive)

I think the answer is “no”

» Resilience of IaC tools

Only a matter of replication?

1. At the Edge or in constrained CPS, disconnection is the norm
   * any distant central node is regularly unreachable (but still alive)
2. Facing extreme climatic events, the edge of the network could be highly impacted, resulting in potentially long network partitioning

I think the answer is “no”

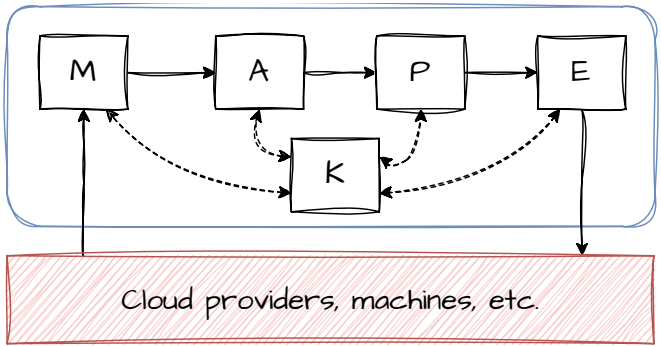
» Resilience of IaC tools

Only a matter of replication?

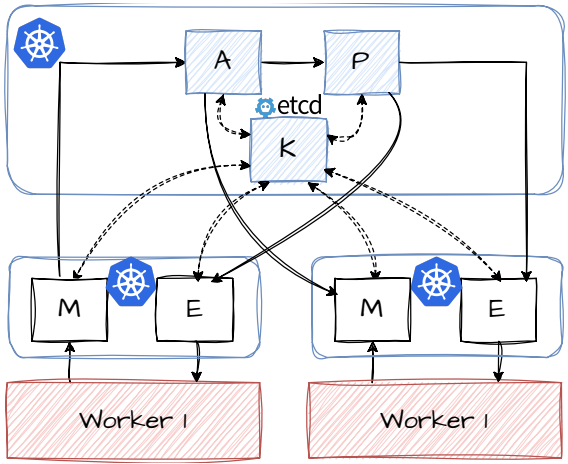
1. At the Edge or in constrained CPS, disconnection is the norm
   * any distant central node is regularly unreachable (but still alive)
2. Facing extreme climatic events, the edge of the network could be highly impacted, resulting in potentially long network partitioning
3. In Cross-DevOps organizations [2], a central state and management is unwanted (security, privacy, size)

I think the answer is “no”

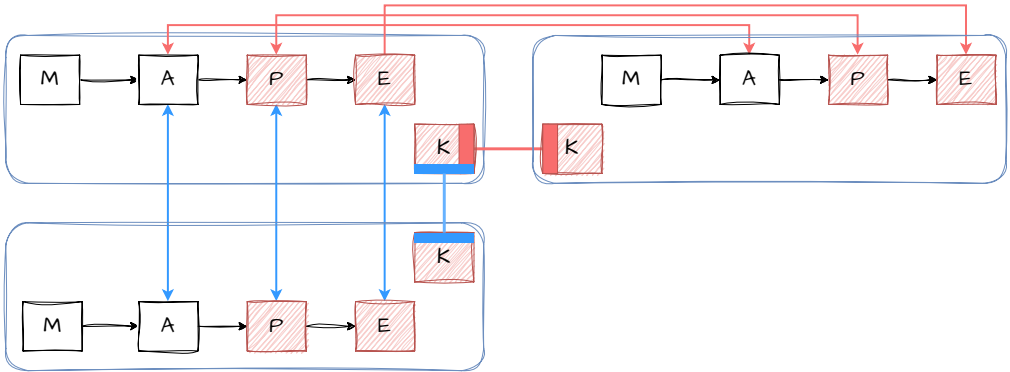
» Decentralized MAPE-K



» Decentralized MAPE-K



» From Concerto to Ballet



Full decentralized MAPE-K: ANR SeMaFoR (WP leader, led by Thomas Ledoux)

Ballet = Decentralized P & E (declarative)

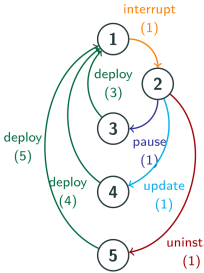
» Concerto-D

Concerto-D vs Concerto

- local `add/del`
- local `con/dcon`
  - communications: synchronized `dcon`
- local `pushB`
  - identified `pushB`
  - communications: statuses of ports
- `wait` for local or distant components
  - communications: end of behaviors

Local progress is possible + less to achieve when the node is back

» Decentralized planner



state	1	2	3	4	5
common	✓	✓	×	×	×
haproxy	×	×	✓	×	×
service	✓	×	×	×	×

Artifacts on Grid'5000

- Local constraint programming model
- Propagation with gossip-like algorithm
- Received messages enrich local models
  - additional behaviors
  - `wait` instructions
- Final consensus: end of the propagation
- Execution is started with Concerto-D

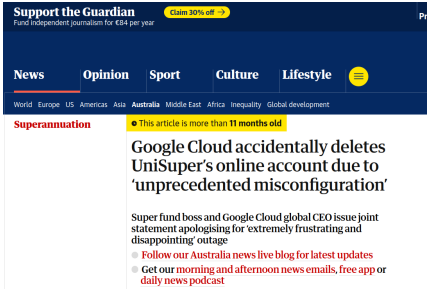
Deployment and update of multi-site OpenStack $[1, 2, 5, 10]$ sites

- 40% (deployment) and 24% (update) gain compared to Mjuz
- Planning time is less than 2% of the execution time
- 10 sites: ∼200 constraints and ∼100 instructions inferred

Safety

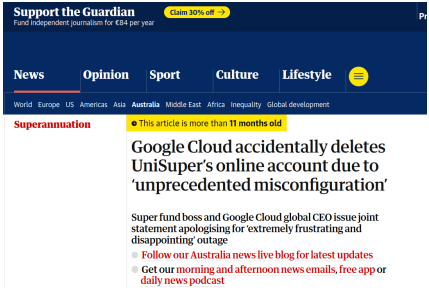
* Why safety?
* Verification
* Formally verified Infrastructure-as-Code

» Why safety?



"Google Cloud CEO, Thomas Kurian has confirmed that the disruption arose from an unprecedented sequence of events whereby an inadvertent misconfiguration during provisioning of UniSuper's Private Cloud services ultimately resulted in the deletion of UniSuper's Private Cloud subscription," the pair said.

» Why safety?



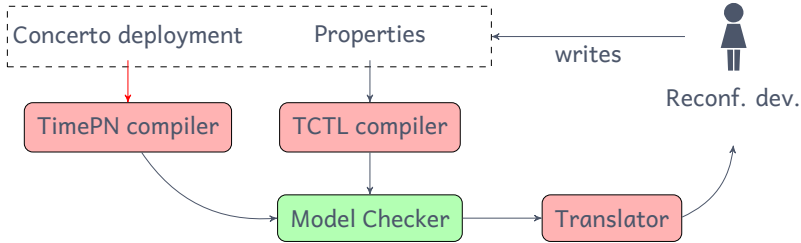
"Google Cloud CEO, Thomas Kurian has confirmed that the disruption arose from an unprecedented sequence of events whereby an inadvertent misconfiguration during provisioning of UniSuper's Private Cloud services ultimately resulted in the deletion of UniSuper's Private Cloud subscription," the pair said.

Services and infrastructures configurations are critical (day-to-day organization, health, energy, network, etc.). Configuration is difficult.

Formal methods: mathematically rigorous techniques for specifying, verifying, and synthesizing software systems

» Verification of a deployment

Restricted to initial deployments



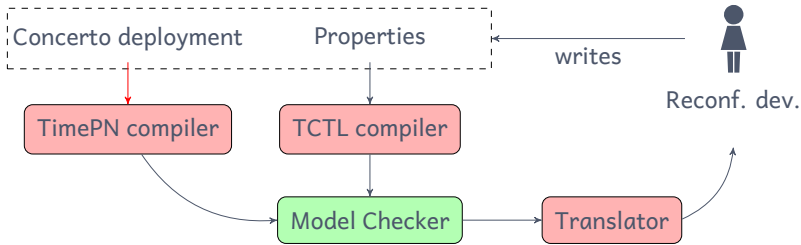
Qualitative properties

- `deployability` (inevitability)
- `sequentiality` (observer subnet)
- `forbidden` (observer subnet)

+ quantitative properties

» Verification of a deployment

Restricted to initial deployments



Qualitative properties

- `deployability` (inevitability)
- `sequentiality` (observer subnet)
- `forbidden` (observer subnet)

+ quantitative properties

Scalablity issues

- Unfeasible for (full) Concerto programs?
- Same issue with FOL and SMT

→ synthesizing correct programs (declarativity)

» Mechanized semantics of Concerto and Concerto-D

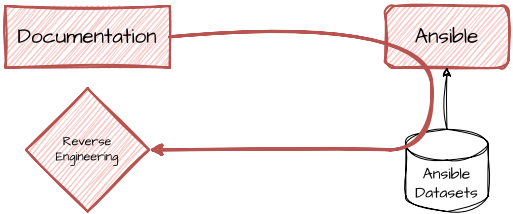
Formal specification of the language (instead of a program)

- Ambiguities (problems) can be discovered and resolved in the language
- Mathematical reference to guide development
- Basis to reason and verify more general properties (any program)

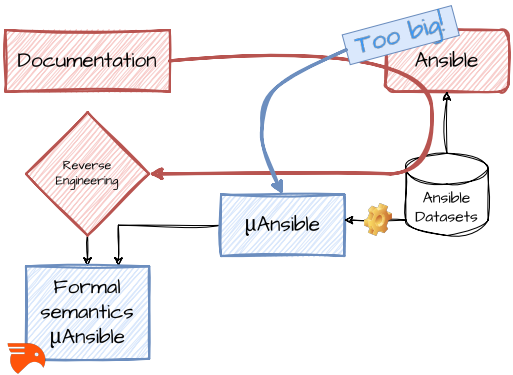
Operational semantics of Concerto and Concerto-D

- Pen-and-paper semantics of Concerto
- Mechanized semantics of Concerto/Concerto-D
  - Maude rewriting system
  - Going to ITP

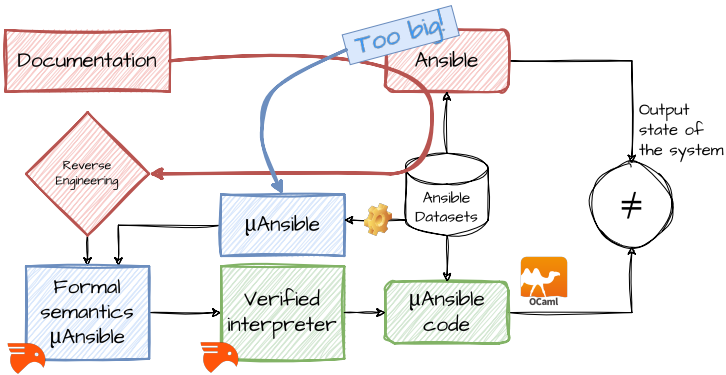
» ANR For-CoaLa



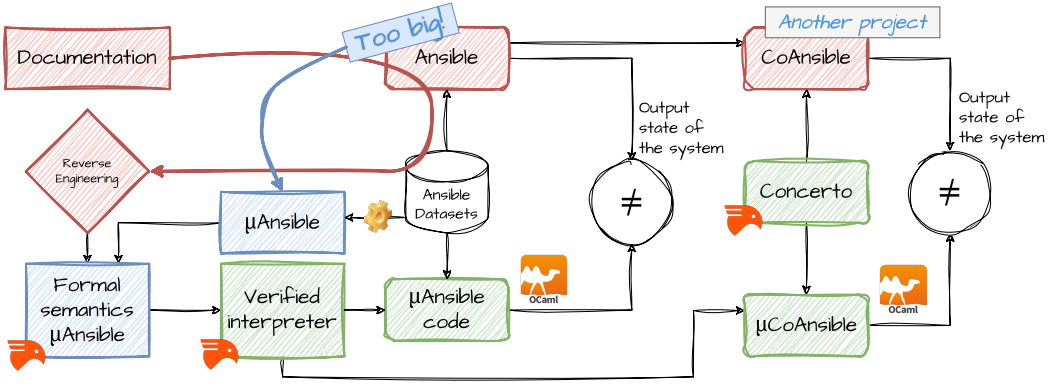
» ANR For-CoaLa



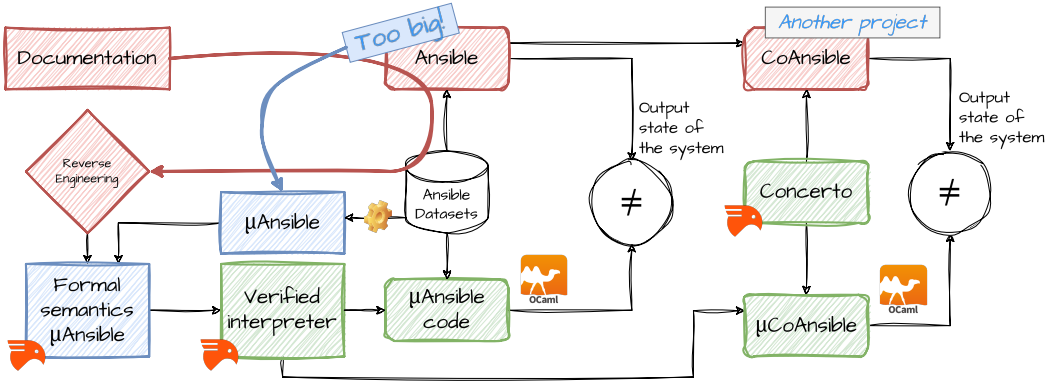
» ANR For-CoaLa



» ANR For-CoaLa



» ANR For-CoaLa



Then verify general theorems on Ansible and CoAnsible![2]

[2] https://for-coala.github.io/about/

Opening

» ICTs are critical infrastructures

ICTs (digital infrastructures) are critical infrastructures

- Other critical infrastructures now rely on ICTs: energy, health, telecommunications, etc.
- ICTs are more and more prominent in our daily life: banks, industries, etc.

» ICTs are critical infrastructures

ICTs (digital infrastructures) are critical infrastructures

- Other critical infrastructures now rely on ICTs: energy, health, telecommunications, etc.
- ICTs are more and more prominent in our daily life: banks, industries, etc.

Vulnerabilities and crises

Like all critical infrastructures, ICTs face vulnerabilities and crises that have to be studied, mitigated, and avoided

» ICTs are critical infrastructures

ICTs (digital infrastructures) are critical infrastructures

- Other critical infrastructures now rely on ICTs: energy, health, telecommunications, etc.
- ICTs are more and more prominent in our daily life: banks, industries, etc.

Vulnerabilities and crises

Like all critical infrastructures, ICTs face vulnerabilities and crises that have to be studied, mitigated, and avoided

Which crises?

» Which crises?

- Climate change
  - humidity, fired, heat/cold waves, flooding, etc.
- Depletion of natural resources
  - energy sources, water, metals
- (Geo)political instability
  - access to natural resources, hardware, software

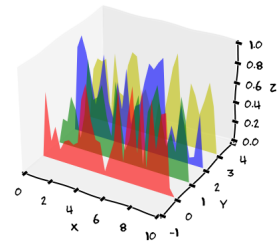
» Fluctuating resources

What these crises have in common

Fluctuation of resources

Which resources?

- energy resources
- network-specific resources
  - connectivity, frequencies
- hardware resources
  - CPUs, GPUs, RAM, disk, antennas etc.
- software resources
  - services, operating systems etc.



» IaC in the face of fluctuating resources

- IaC for crises
  - unavailability of part of infrastructures
  - priorities of infrastructure resources
  - stochastic model of resources
  - high heterogeneity of resources
  - etc.
- IaC resilience to crises
  - decentralized IaC
  - local-first IaC
- Safety aspects of resiliency
  - formal properties of resiliency?
    - absorption/mitigation/recovery

» Persons involved in the presented contributions and projects

Efficiency

C. Perez
DR Inria
REP Avalon

D. Pertin
Postdoc

M. Chardet
PhD St.

S. Robillard
Postdoc

J. Philippe
Postdoc

C. Prud'homme
As. Prof.
IMT Atlantique

[ADT Inria CoAnsible]

B.Jonglez
IR Inria

S.M. Kaddour
Soft. engineer

Decentralization

I. Raïs
As. Prof
UiT Norway

A. Omond
PhD st.

J. Philippe
Postdoc

PhDs
Postdocs

Safety

[VeRDi]

M. Chardet
PhD St.

S. Robillard
Postdoc

[PIA OTPaaS]

F. Arfi
Postdoc

L. Henrio
CR CNRS

F. Loulergue
Prof.
Univ. Orléans

C. Jard
Prof.
Univ. Nantes

D. Lime
Prof.
Centrale Nantes

[ANR For-CoaLa]

F. Loulergue
Prof.
Univ. Orléans

O. Proust
PhD St.

» References

[1] J.O. Kephart and D.M. Chess.
The vision of autonomic computing.
Computer, 36(1):41–50, 2003.

[2] Daniel Sokolowski, Pascal Weisenburger, and Guido Salvaneschi.
Automating serverless deployments for devops organizations.
In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA, 2021. Association for Computing Machinery.

[3] Rosemary Wang.
Infrastructure as Code, Patterns and Practices: With Examples in Python and Terraform.
Simon and Schuster, 2022.

Mentioned contributions

- Jolan Philippe, Antoine Omond, Hélène Coullon, Charles Prud'homme, Issam Rais. *Fast Choreography of Cross-DevOps Reconfiguration with Ballet: A Multi-Site OpenStack Case Study*. In IEEE SANER 2024, Finland
- Farid Arfi, Hélène Coullon, Frédéric Loulergue, Jolan Philippe, Simon Robillard. A Maude Formalization of the Distributed Reconfiguration Language Concerto-D. In ICE@DisCoTeC 2024, Groningen, The Netherlands
- Simon Robillard, Hélène Coullon. *SMT-Based Planning Synthesis for Distributed System Reconfigurations*. In FASE 2022, Munich, Germany
- Maverick Chardet, Hélène Coullon, Simon Robillard. *Toward Safe and Efficient Reconfiguration with Concerto*. SCP, 2021
- Maverick Chardet, Hélène Coullon, Christian Perez. *Predictable Efficiency for Reconfiguration of Service-Oriented Systems with Concerto*. In CCGrid 2020, Melbourne, Australia
- Hélène Coullon, Didier Lime, Claude Jard. *Integrated Model-checking for the Design of Safe and Efficient Distributed Software Commissioning*. In iFM 2019, Bergen, Norway

» Thank you for your attention!

Infrastructure-as-Code
Efficiency
Decentralization
Safety
Opening