Compiling Abstract Specifications into Concrete Systems – Bringing Order to the Cloud

ANCOR

- **A**utomated e**N**terprise network **CO**mpile**R** -

Ian Unruh, **Alexandru G. Bardas**,

Rui Zhuang, Xinming Ou, Scott DeLoach

KANSAS STATE UNIVERSITY

Cloud Users - Desired Features

- **Flexibility**
  - access to the raw resources *e.g.,* compute, storage
- Reliable **automation** capabilities
  - Non-scenario-dependent
  - Automatic deployment and maintenance
  - Dynamic cluster expansion and contraction
- **Migration** between different could providers
  - Capturing infrastructure and application requirements in a specification

Current Cloud Computing Offerings

- Allow customers to decide ***how much*** **management they want:**
  - Infrastructure as a Service (IaaS)
    - *e.g.,* Amazon Web Services, OpenStack

  - Platform as a Service (PaaS)
    - *e.g.,* Heroku, Microsoft Azure

  - Software as a Service (SaaS)
    - *e.g.,* SalesForce, Google Apps

Current Cloud Computing Offerings

- Allow customers to decide ***how much*** **management they want:**
  - Infrastructure as a Service (IaaS)
    - *e.g.,* Amazon Web Services, OpenStack

✔	**Flexibility** (access to the raw resources)
✖	**Automation** (non-scenario-dependent)
✖	**Migration** (capturing requirements in a specification)

Current Cloud Computing Offerings

- Allow customers to decide ***how much*** **management they want:**
  - Software as a Service (SaaS)
    - *e.g.,* SalesForce, Google Apps

✖	**Flexibility** (access to the raw resources)
✔	**Automation** (non-scenario-dependent)
✖	**Migration** (capturing requirements in a specification)

Current Cloud Computing Offerings

- Allow customers to decide ***how much*** **management they want:**
  - Platform as a Service (PaaS)
    - *e.g.,* Heroku, Microsoft Azure

✖	**Flexibility** (access to the raw resources)
✔	**Automation** (non-scenario-dependent)
✖	**Migration** (capturing requirements in a specification)

Proposed Solution

- An **abstraction** that captures ***what*** a cloud user needs instead of low-level details on ***how*** to implement those needs

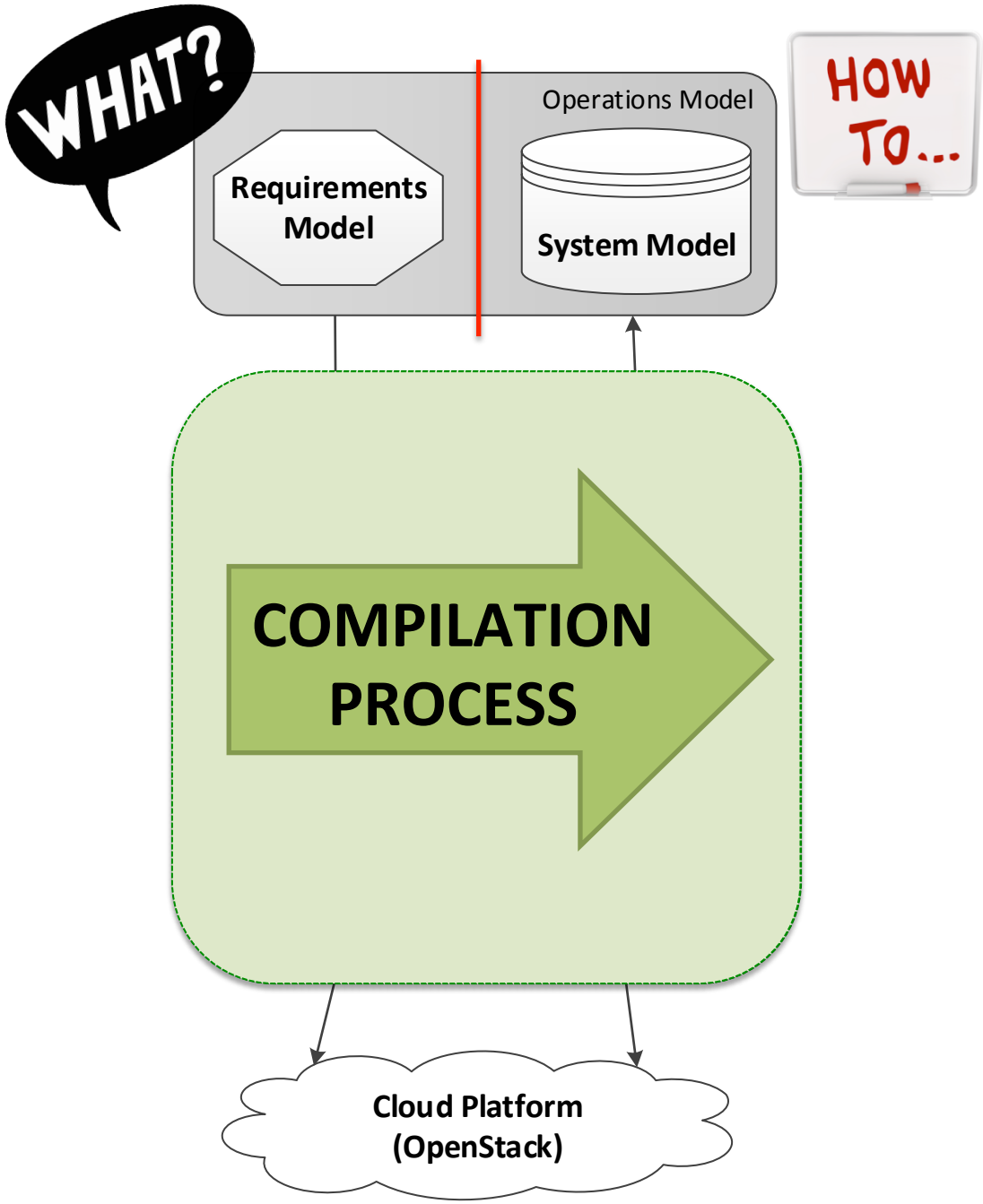
- There must be a process to **automatically compile the abstraction into a valid concrete system**

Proposed Solution

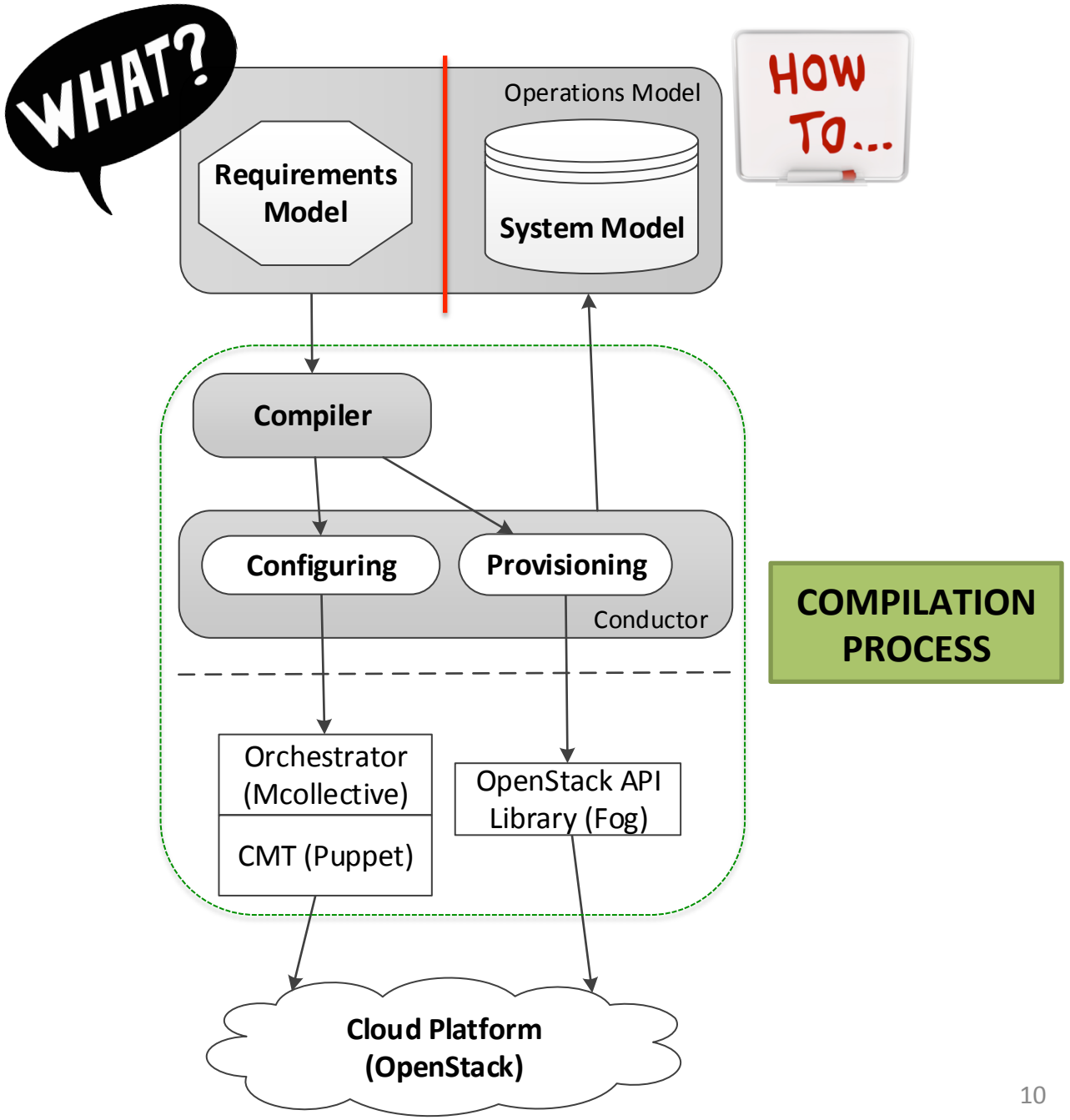
- An **abstraction** that captures ***what*** a cloud user needs instead of low-level details on ***how*** to implement those needs

- There must be a process to **automatically compile the abstraction into a valid concrete system**

ANCOR

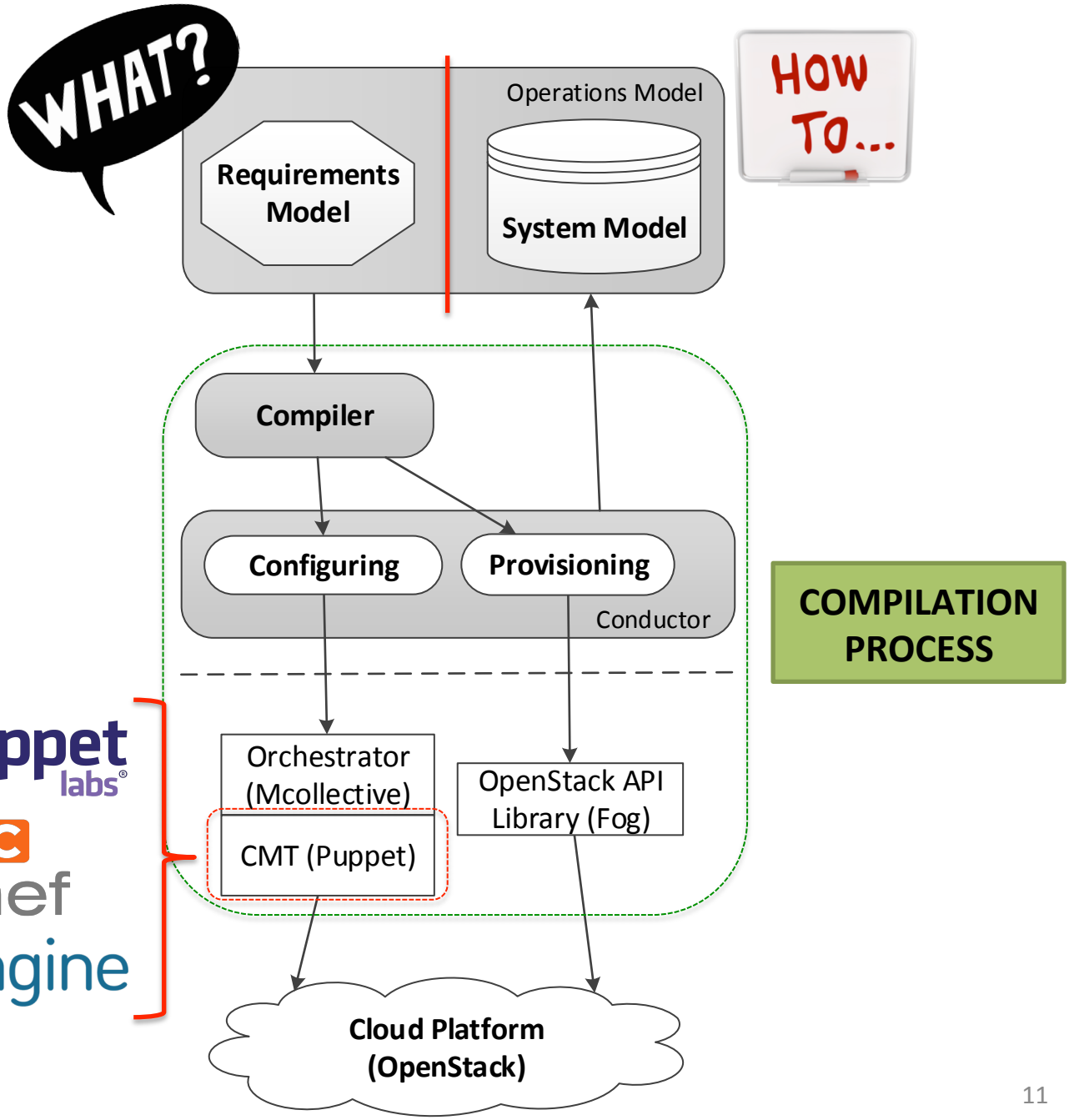

WHAT?
Operations Model
Requirements Model
System Model
HOW TO...
COMPILATION PROCESS
Cloud Platform (OpenStack)

ANCOR

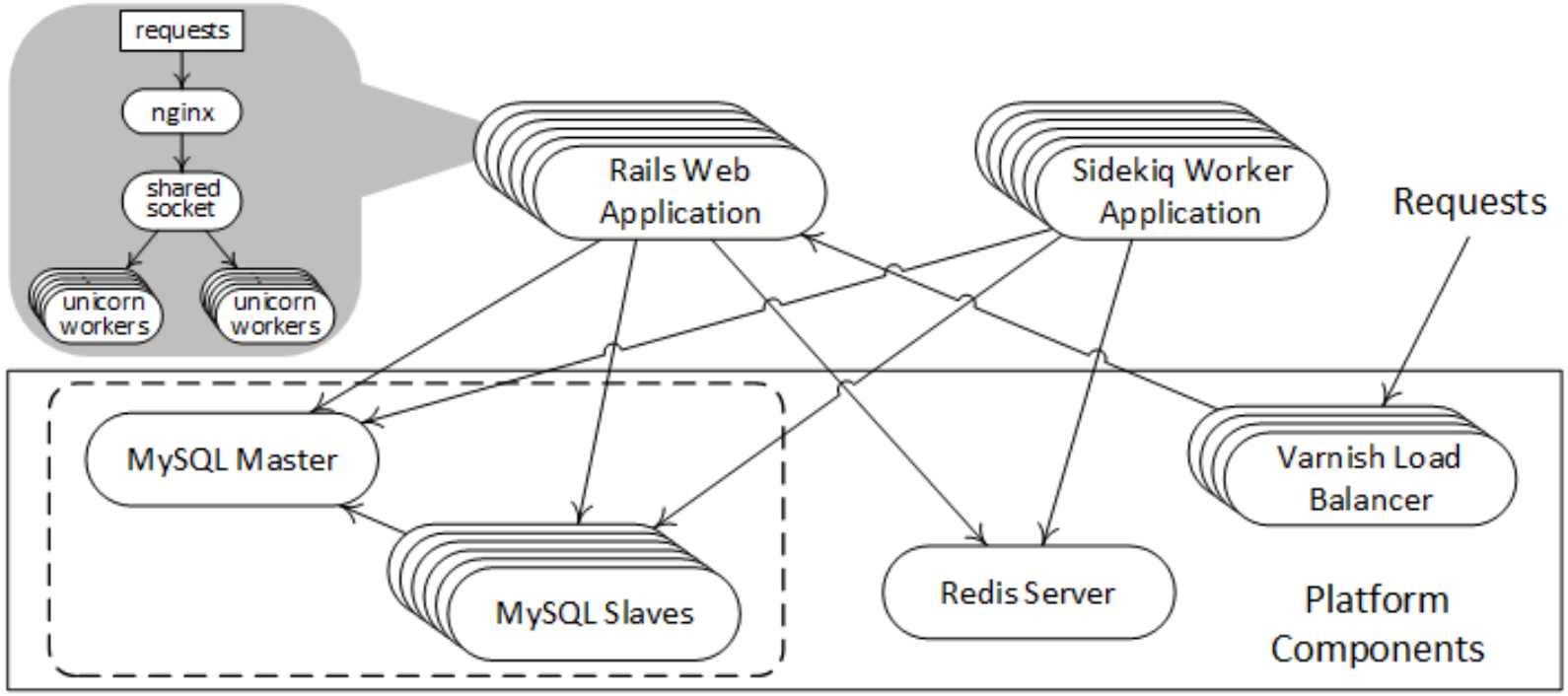


COMPILATION PROCESS

ANCOR


WHAT?
HOW TO...
Operations Model
Requirements Model
System Model
Compiler
Configuring
Provisioning
Conductor
COMPILATION PROCESS
Orchestrator (Mcollective)
CMT (Puppet)
OpenStack API Library (Fog)
Cloud Platform (OpenStack)
Configuration Management Tools (CMTs)
puppet labs
Chef
CFEngine

Deploying an eCommerce Website



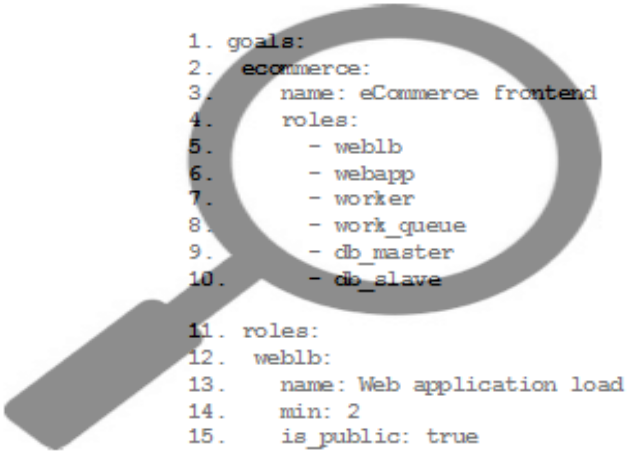
Scalable and highly available eCommerce website architecture

ANCOR Requirement Modeling Language (ARML)

```
goals:
 ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

roles:
 weblb:
    name: Web application load balancer
    min: 2
    is_public: true
    implementations:
      default:
        profile: role::weblb::default
    exports:
      http:{ type: single_port, protocol: tcp, number:80 }
    imports:
      webapp: http

 webapp:
    name: Web application
    min: 3
    implementations:
      default:
        profile: role::webapp::default
    exports:
      http: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

worker:
  name: Sidekiq worker application
  min: 2
  implementations:
    default:
      profile: role::worker::default
  imports:
    db_master: querying
    db_slave: querying
    work_queue: redis

work_queue:
  name: Redis work queue
  implementations:
    default:
      profile: role::work_queue::default
  exports:
    redis: { type: single_port, protocol: tcp }

db_master:
  name: MySQL master
  implementations:
    default:
      profile: role::db_master::default
  exports:
    querying: { type: single_port, protocol: tcp }

db_slave:
  name: MySQL slave
  implementations:
    default:
      profile: role::db_slave::default
  min: 2
  exports:
    querying: { type: single_port, protocol: tcp }
  imports:
    db_master: querying
```

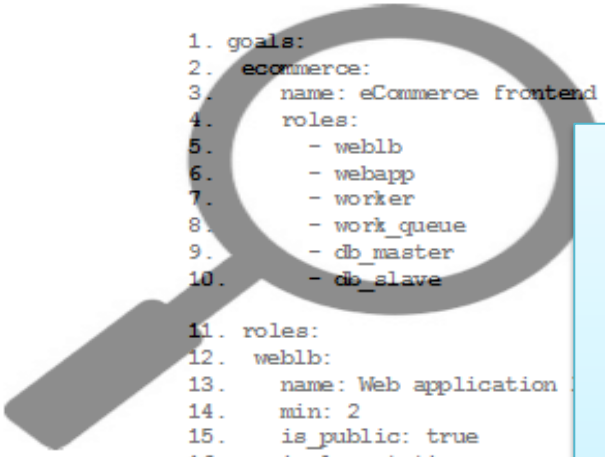
eCommerce Website Requirements Specification

Ancor Requirement Modeling Language (ARML)



```
goals:
  ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

roles:
  weblb:
    name: Web application load balancer
    min: 2
    is_public: true
    implementations:
      default:
        profile: role::weblb::default
    exports:
      http:{ type: single_port, protocol: tcp, number:80 }
    imports:
      webapp: http

  webapp:
    name: Web application
    min: 3
    implementations:
      default:
        profile: role::webapp::default
    exports:
      http: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

worker:
  name: Sidekiq worker application
  min: 2
  implementations:
    default:
      profile: role::worker::default
  imports:
    db_master: querying
    db_slave: querying
    work_queue: redis

work_queue:
  name: Redis work queue
  implementations:
    default:
      profile: role::work_queue::default
  exports:
    redis: { type: single_port, protocol: tcp }

db_master:
  name: MySQL master
  implementations:
    default:
      profile: role::db_master::default
  exports:
    querying: { type: single_port, protocol: tcp }

db_slave:
  name: MySQL slave
  implementations:
    default:
      profile: role::db_slave::default
  min: 2
  exports:
    querying: { type: single_port, protocol: tcp }
  imports:
    db_master: querying
```

Ancor Requirement Modeling Language (ARML)

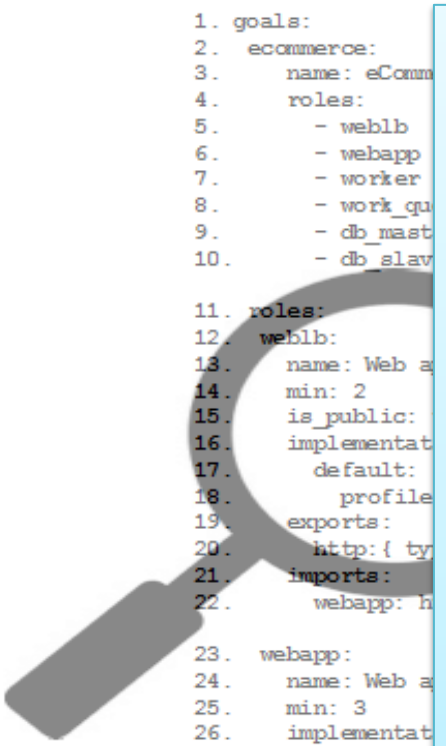


```
goals:
  ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave
```

Ancor Requirement Modeling Language (ARML)

```
goals:
 ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

roles:
 weblb:
    name: Web application load balancer
    min: 2
    is_public: true
    implementations:
      default:
        profile: role::weblb::default
    exports:
      http:{ type: single_port, protocol: tcp, number:80 }
    imports:
      webapp: http

 webapp:
    name: Web application
    min: 3
    implementations:
      default:
        profile: role::webapp::default
    exports:
      http: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

worker:
 name: Sidekiq worker application
 min: 2
 implementations:
   default:
     profile: role::worker::default
 imports:
   db_master: querying
   db_slave: querying
   work_queue: redis

work_queue:
 name: Redis work queue
 implementations:
   default:
     profile: role::work_queue::default
 exports:
   redis: { type: single_port, protocol: tcp }

db_master:
 name: MySQL master
 implementations:
   default:
     profile: role::db_master::default
 exports:
   querying: { type: single_port, protocol: tcp }

db_slave:
 name: MySQL slave
 implementations:
   default:
     profile: role::db_slave::default
 min: 2
 exports:
   querying: { type: single_port, protocol: tcp }
 imports:
   db_master: querying
```

Ancor Requirement Modeling Language (ARML)

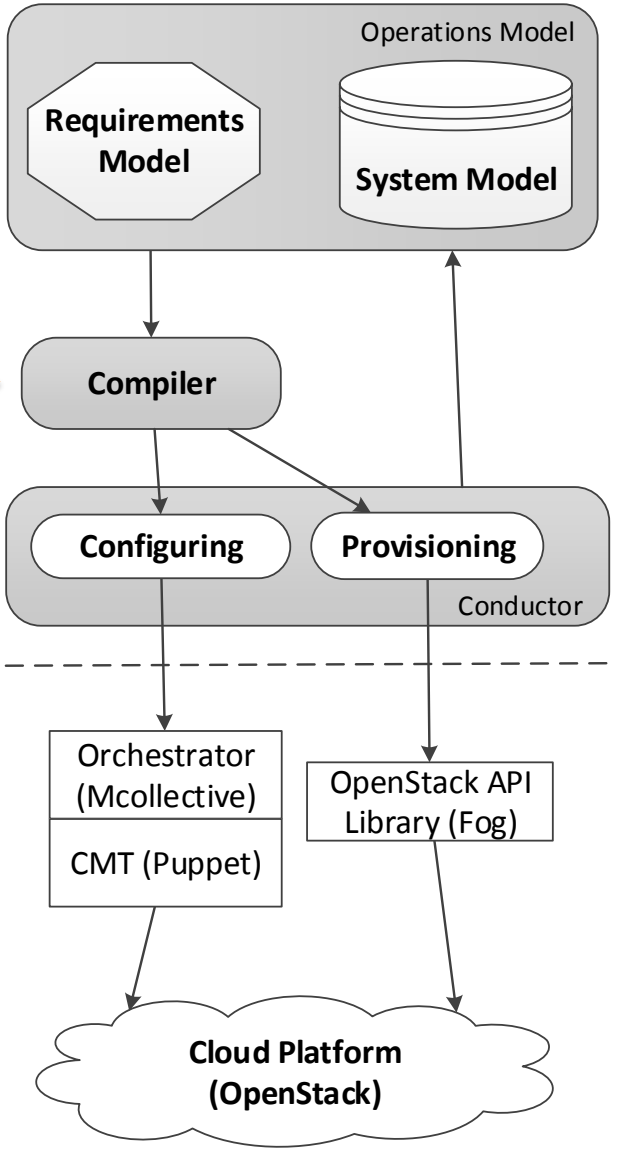


```
roles:
  weblb:
    name: Web application load balancer
    min: 2
    is_public: true
    implementations:
      default:{ profile:
                  "role::ecommerce::
                   weblb::default" }
    exports:
      http: { type: single_port,
              protocol: tcp, number: 80 }
    imports:
      webapp: http
```

ANCOR Workflow

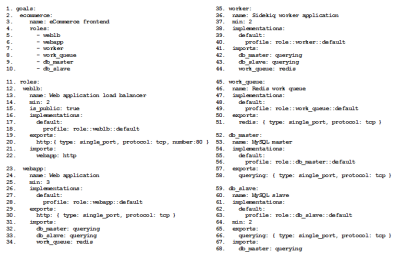
```
goals:
  ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

roles:
  weblb:
    name: Web application load balancer
    min: 2
    ha_public: true
    implementations:
      default:
        profile: role::weblb::default
    exports:
      http: { type: single_port, protocol: tcp, number:80 }
    imports:
      webapp: http

  webapp:
    name: Web application
    min: 3
    implementations:
      default:
        profile: role::webapp::default
    exports:
      http: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  worker:
    name: Sidekiq worker application
    min: 2
    implementations:
      default:
        profile: role::worker::default
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  work_queue:
    name: Redis work queue
    implementations:
      default:
        profile: role::work_queue::default
    exports:
      redis: { type: single_port, protocol: tcp }

  db_master:
    name: MySQL master
    implementations:
      default:
        profile: role::db_master::default
    exports:
      querying: { type: single_port, protocol: tcp }

  db_slave:
    name: MySQL slave
    implementations:
      default:
        profile: role::db_slave::default
    min: 2
    exports:
      querying: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
```

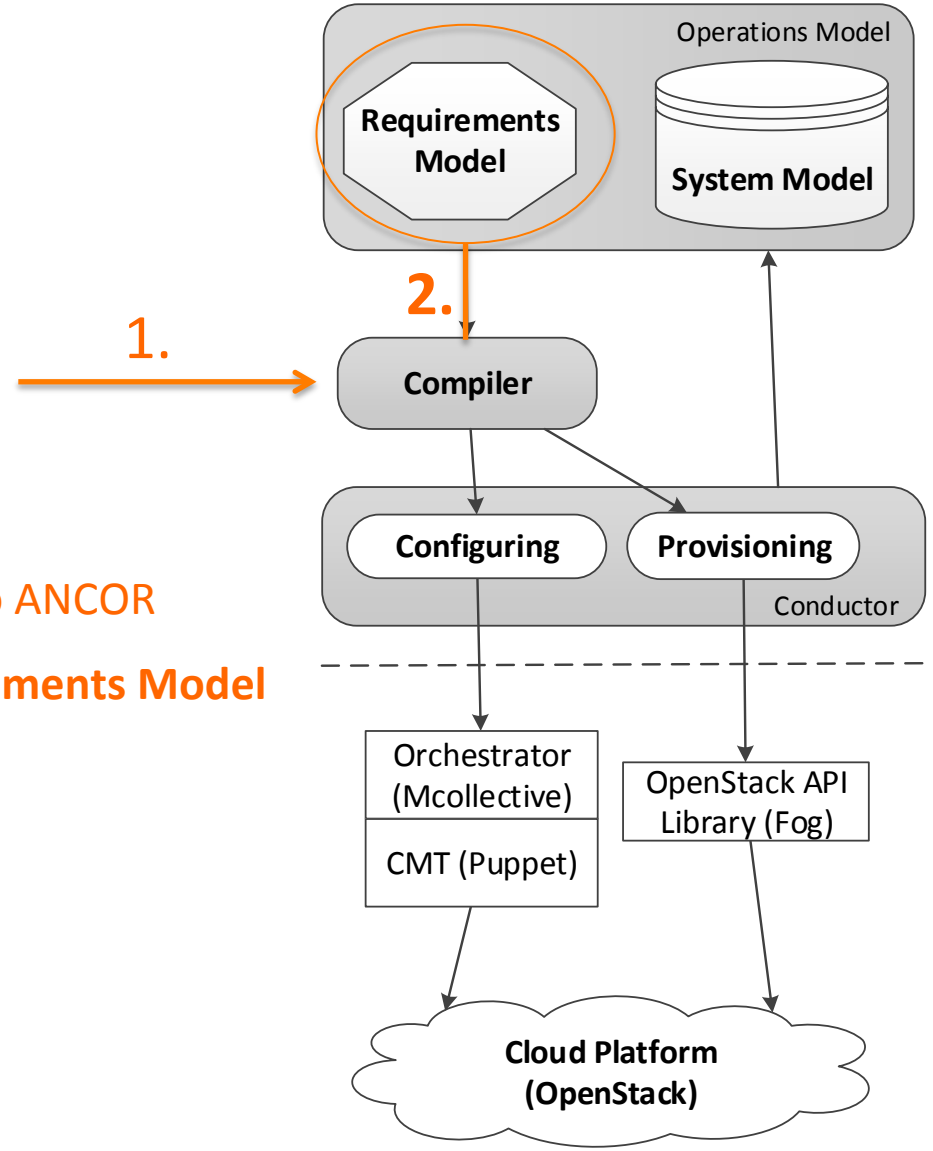
1.



1. **Passing the IT system specification to ANCOR**

ANCOR Workflow

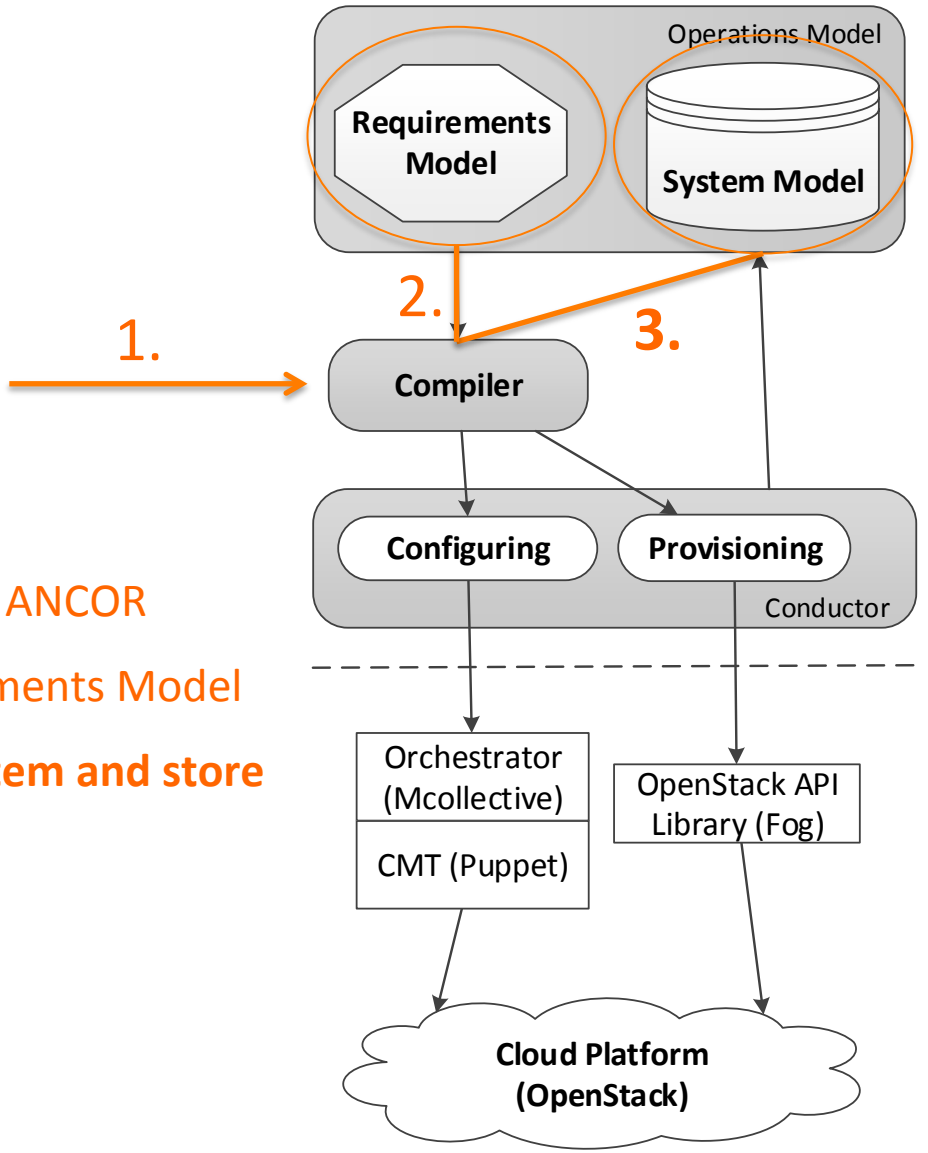




1. Passing the IT system specification to ANCOR
2. **Specification is stored in the Requirements Model**

ANCOR Workflow

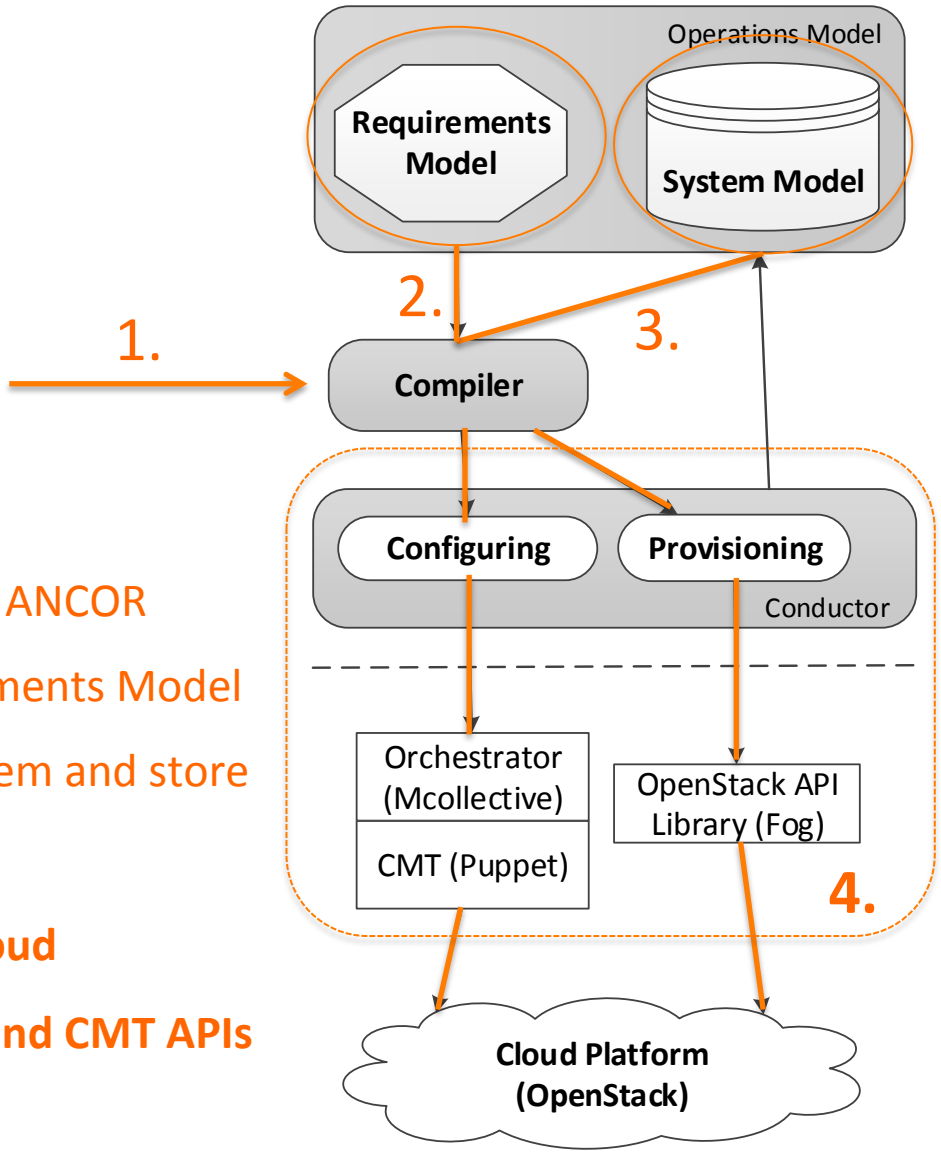
```
goals:
  ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

roles:
  weblb:
    name: Web application load balancer
    min: 2
    ha_public: true
    implementations:
      default:
        profile: role::weblb::default
    exports:
      http: { type: single_port, protocol: tcp, number:80 }
    imports:
      webapp: http

  webapp:
    name: Web application
    min: 3
    implementations:
      default:
        profile: role::webapp::default
    exports:
      http: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  worker:
    name: Sidekiq worker application
    min: 2
    implementations:
      default:
        profile: role::worker::default
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  work_queue:
    name: Redis work queue
    implementations:
      default:
        profile: role::work_queue::default
    exports:
      redis: { type: single_port, protocol: tcp }

  db_master:
    name: MySQL master
    implementations:
      default:
        profile: role::db_master::default
    exports:
      querying: { type: single_port, protocol: tcp }

  db_slave:
    name: MySQL slave
    implementations:
      default:
        profile: role::db_slave::default
    min: 2
    exports:
      querying: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
```



1. Passing the IT system specification to ANCOR
2. Specification is stored in the Requirements Model
3. **Compute low-level details of the system and store them in the System Model**

ANCOR Workflow

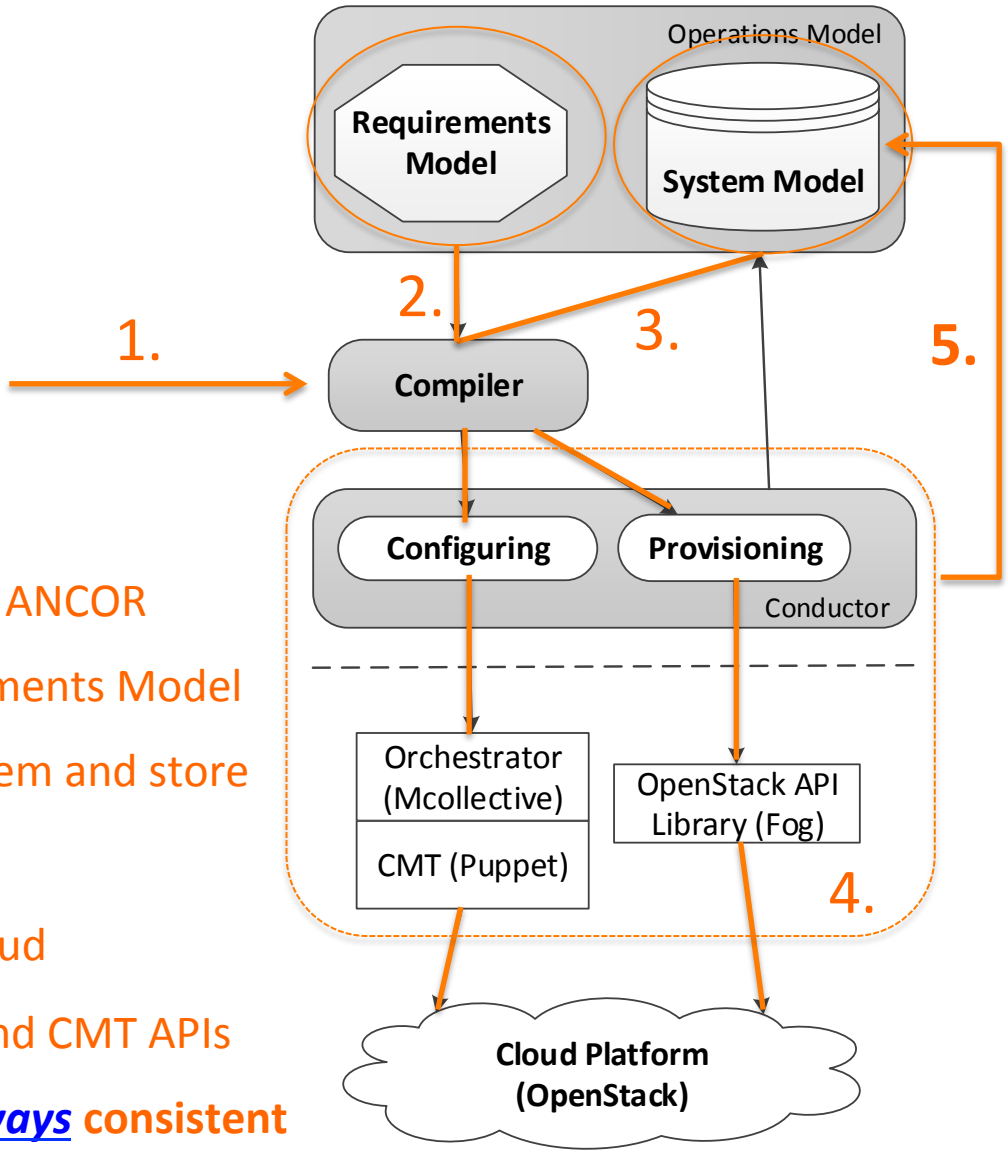
```
goals:
  ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

roles:
  weblb:
    name: Web application load balancer
    min: 2
    ha_public: true
    implementations:
      default:
        profile: role::weblb::default
    exports:
      http: { type: single_port, protocol: tcp, number:80 }
    imports:
      webapp: http

  webapp:
    name: Web application
    min: 3
    implementations:
      default:
        profile: role::webapp::default
    exports:
      http: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  worker:
    name: Sidekiq worker application
    min: 2
    implementations:
      default:
        profile: role::worker::default
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  work_queue:
    name: Redis work queue
    implementations:
      default:
        profile: role::work_queue::default
    exports:
      redis: { type: single_port, protocol: tcp }

  db_master:
    name: MySQL master
    implementations:
      default:
        profile: role::db_master::default
    exports:
      querying: { type: single_port, protocol: tcp }

  db_slave:
    name: MySQL slave
    implementations:
      default:
        profile: role::db_slave::default
    min: 2
    exports:
      querying: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
```



1. Passing the IT system specification to ANCOR
2. Specification is stored in the Requirements Model
3. Compute low-level details of the system and store them in the System Model
4. **Start deploying the system on the cloud infrastructure using cloud platform and CMT APIs**

ANCOR Workflow

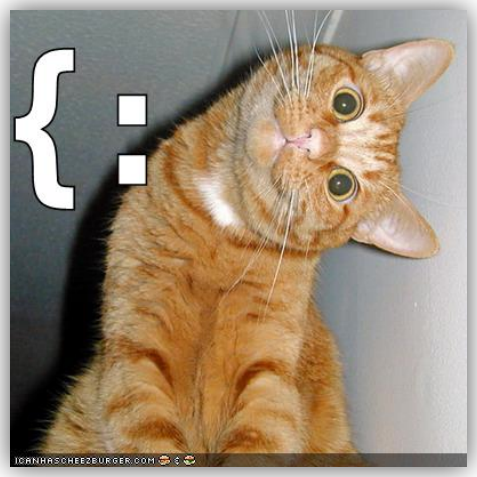
```
goals:
  ecommerce:
    name: eCommerce frontend
    roles:
      - weblb
      - webapp
      - worker
      - work_queue
      - db_master
      - db_slave

roles:
  weblb:
    name: Web application load balancer
    min: 2
    public: true
    implementations:
      default:
        profile: role::weblb::default
    exports:
      http: { type: single_port, protocol: tcp, number:80 }
    imports:
      webapp: http

  webapp:
    name: Web application
    min: 3
    implementations:
      default:
        profile: role::webapp::default
    exports:
      http: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  worker:
    name: Sidekiq worker application
    min: 2
    implementations:
      default:
        profile: role::worker::default
    imports:
      db_master: querying
      db_slave: querying
      work_queue: redis

  work_queue:
    name: Redis work queue
    implementations:
      default:
        profile: role::work_queue::default
    exports:
      redis: { type: single_port, protocol: tcp }

  db_master:
    name: MySQL master
    implementations:
      default:
        profile: role::db_master::default
    exports:
      querying: { type: single_port, protocol: tcp }

  db_slave:
    name: MySQL slave
    implementations:
      default:
        profile: role::db_slave::default
    min: 2
    exports:
      querying: { type: single_port, protocol: tcp }
    imports:
      db_master: querying
```



1. Passing the IT system specification to ANCOR
2. Specification is stored in the Requirements Model
3. Compute low-level details of the system and store them in the System Model
4. Start deploying the system on the cloud infrastructure using cloud platform and CMT APIs
5. **Update the System Model so it is *always* consistent with the deployed system**

ANCOR Benefits

- ✓ Infrastructure and application “portability”
- ✓ Up-to-date application dependencies
- ✓ Building highly dynamic systems
- ✓ Automated fine-grained firewall configuration
- ✓ Security assessments
- ✓ Performance evaluations
- ✓ Creating customized PaaS



ANCOR

- Current implementation and more information:

http://arguslab.github.io/ancor/

Conclusion

- Separating user requirements from the implementation details has the potential of changing the way IT systems are deployed and managed in the cloud
- ANCOR – framework that captures the high-level requirements and translates them into a working IT system on a cloud infrastructure

http://arguslab.github.io/ancor/

LISA Labs

Today 4:00PM – 5:30PM

Alex Bardas: bardasag@ksu.edu

Image Source: http://tinyurl.com/pr5n8gz

Related Work

- ***Automation Solutions***
  - Automating instance management *e.g.,* AWS OpsWorks
  - Deploying/migrating applications on different cloud providers *e.g.*, Cliqr, Cloud Velocity, CloudSwitch
  - Managing and automating instances deployment *e.g.*, Right-Scale, Service-Now
- ***Abstraction Approaches*** (PaaS specific)
  - Windows Azure Service Definition Schema (.csdef), Google AppEngine YAML-based specification
- ***Managing Infrastructure*** (support CMT integration)
  - OpenStack Heat, AWS CloudFormation, Terraform

More Related Work

- Docker container-based solutions:
  - Maestro-NG, Flynn, Deis, OpenShift, *etc.*

- Ubuntu Juju:
  - Works at a similar abstraction level

ANCOR vs. JUJU

- Similarities:
  - Work at a similar abstraction level
  - Have a way of capturing the dependencies between software applications (services)

- Differences:
  - Using existent CMT modules and workflow:
    - ANCOR: minimal changes
    - Juju: usually, significant changes (integration with Juju Tools)
  - Dependent services
    - ANCOR: more “centralized” management scheme
    - Juju: negotiation scheme
  - Current feature sets *e.g,* OS support