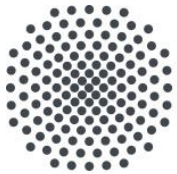


How to Deploy & Manage Distributed Systems Automatically?



University of Stuttgart

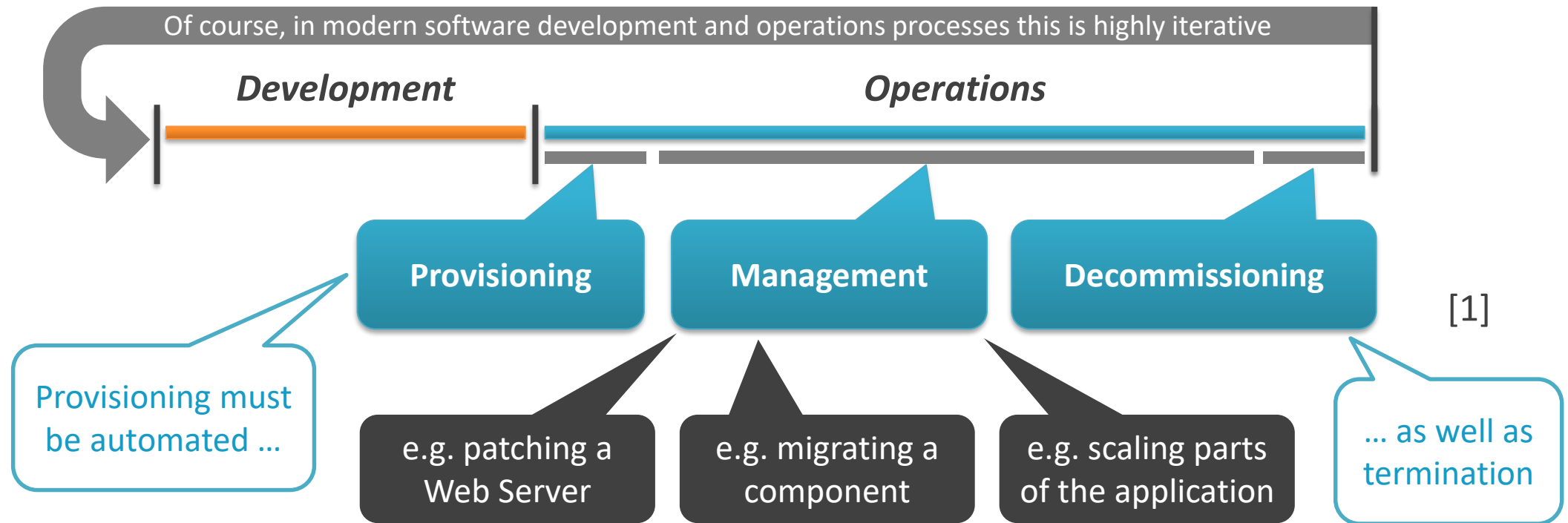
Uwe Breitenbücher

breitenbuecher@iaas.uni-stuttgart.de

Institute of Architecture of Application Systems

Application Lifecycle Management

- Coarse-grained phases of an application's lifecycle



¹ Based on David Chappel. „What is application lifecycle management?“, 2010.

*How to automate the provisioning of
single-component applications?*

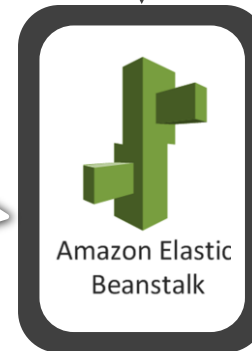
A possible solution: Use a Platform as a Service offering

- We could simply deploy our application on a **Platform as a Service offering**

These offerings natively support ***automatic management features***, e.g., auto scaling

Monitoring System

Auto Scaler



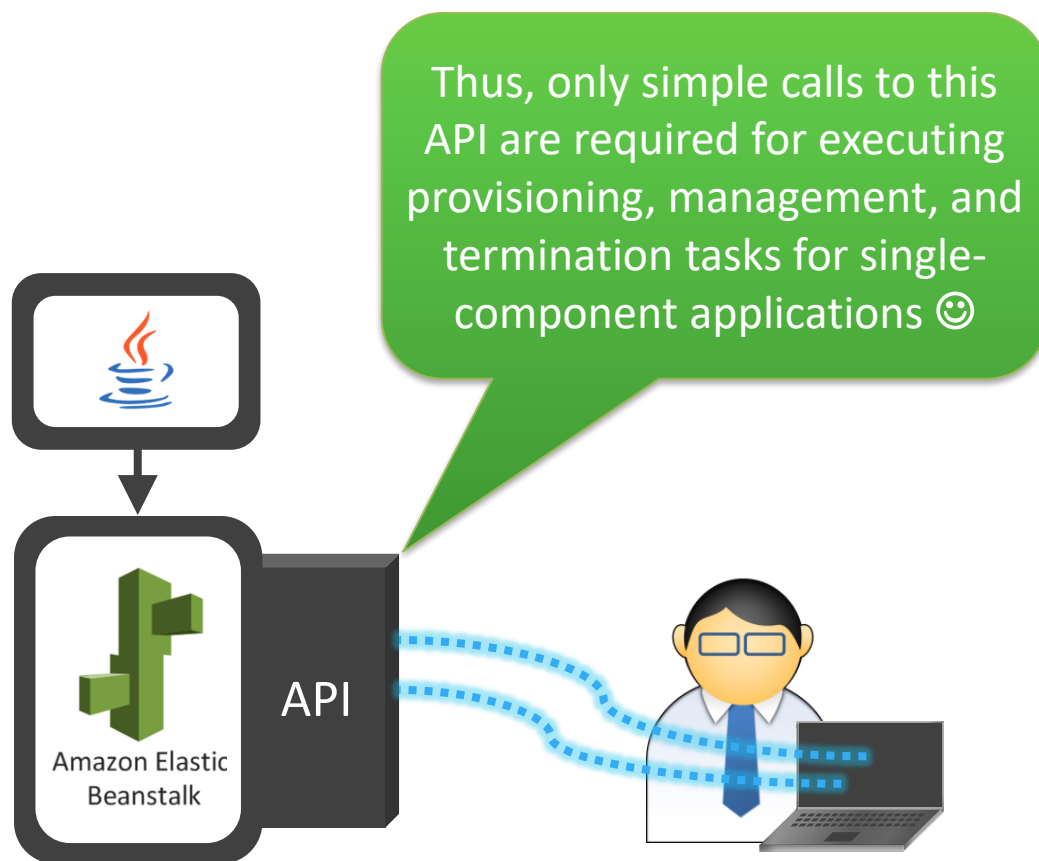
API

... and provide APIs that can be used for ***automated provisioning, management and decommissioning***, e.g., to restart the application



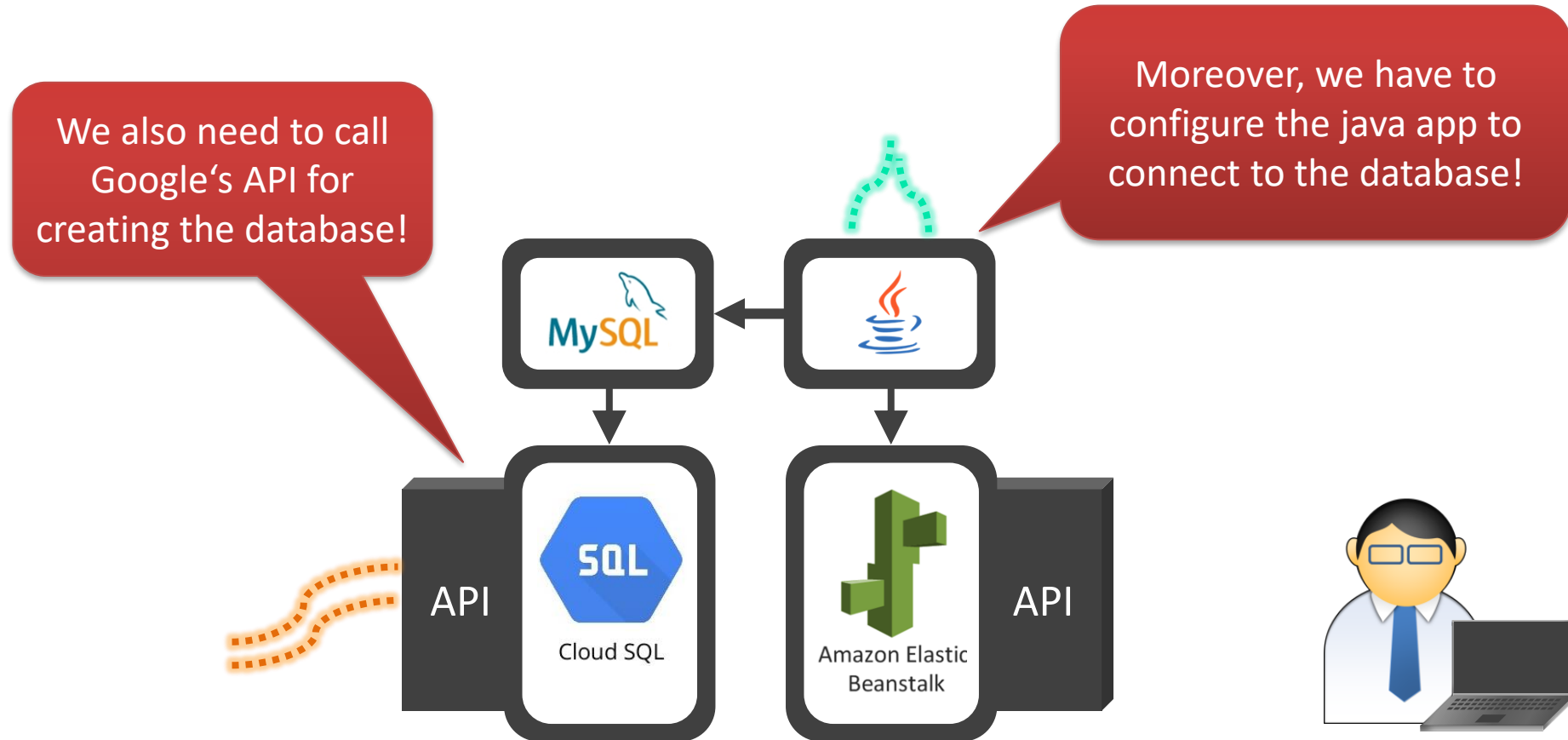
A possible solution: Use a Platform as a Service offering

- We could simply deploy our application on a **Platform as a Service offering**



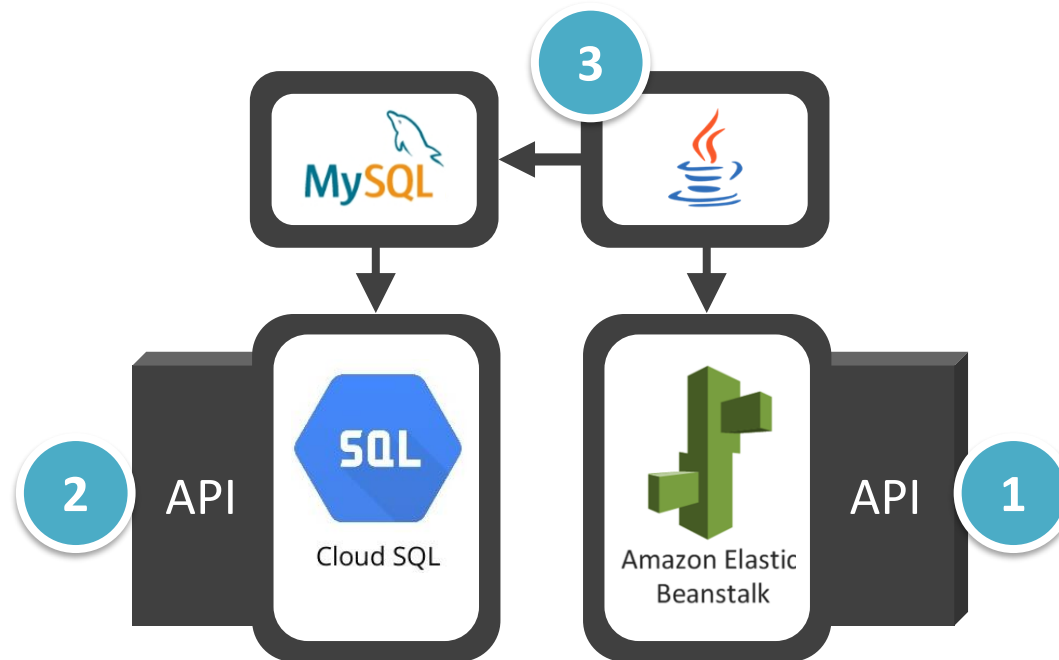
However, what happens if ...

- ... we have a more complex application that requires an external database?



Result

- So in total, we need to execute **3 tasks** for deploying this simple application
 1. Deploy application using Amazon's API
 2. Create database using Google's API
 3. Connect the application to the database using the command line



Result

- So in total, we need to execute **3 tasks** for deploying this simple application
 1. Deploy application using Amazon's API
 2. Create database using Google's API
 3. Connect the application to the database using the command line

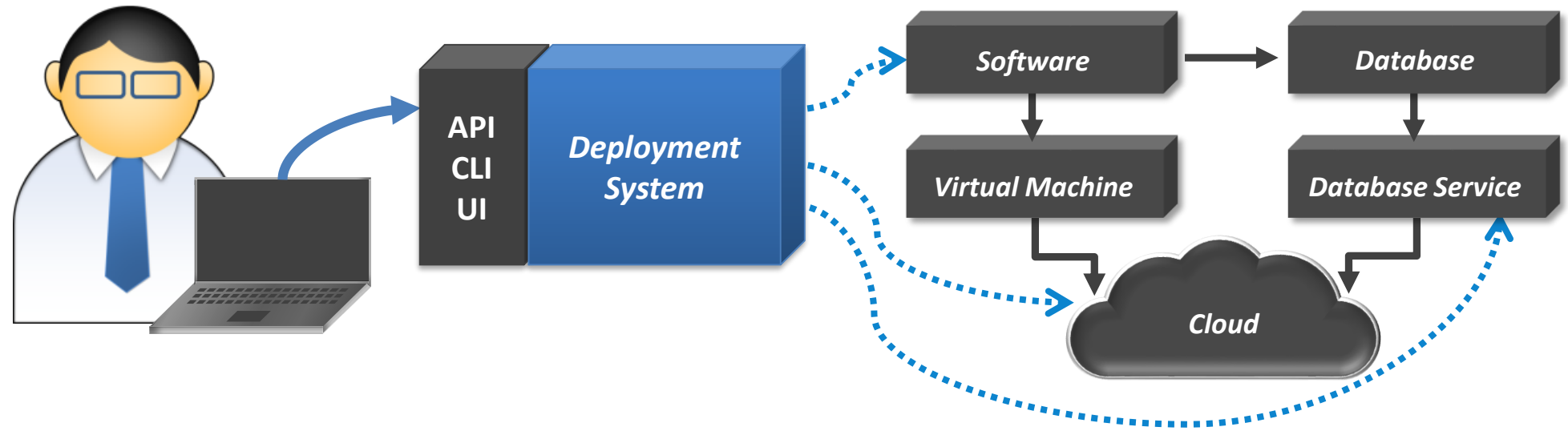
However, manually triggering these 3 tasks breaks automation!

Thus, we need a means to automatically execute more than one task
to provision and manage complex distributed systems
that consist of multiple components

So the interesting question is:

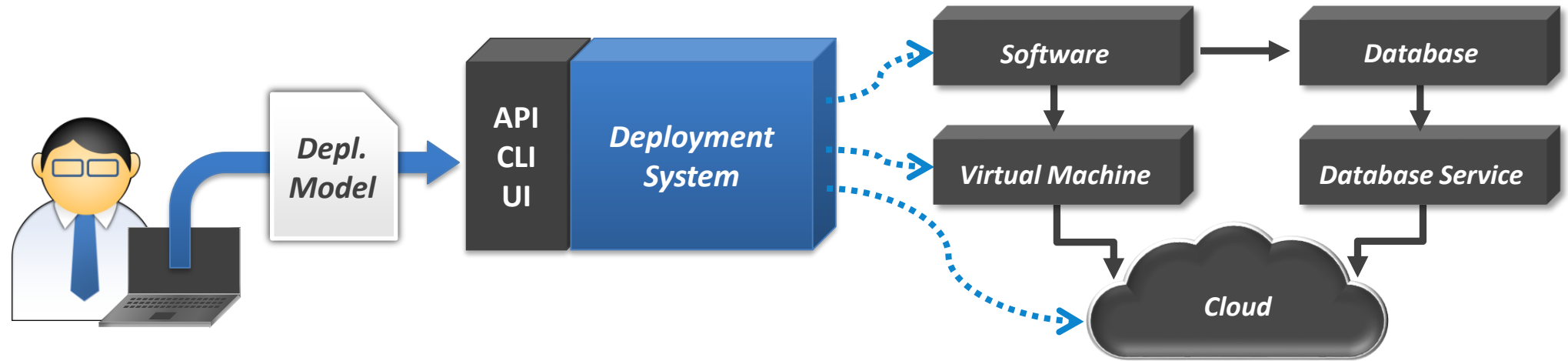
*How to automate the provisioning of
complex distributed systems?*

Deployment Systems



- There are multiple different **deployment systems** for deploying applications in an automated manner
 - Typically these systems provide an API, CLI, or can be controlled via an UI
 - Deployment capabilities range from deploying single virtual machine images to deploying entire composite applications on heterogeneous infrastructures and different kinds of services

Deployment Systems – Model-based deployment

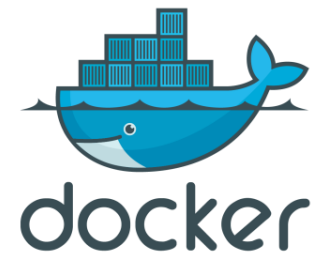


- Therefore, most available deployment systems are ***model-based***:
 - A ***deployment model*** is created by the developer or operations staff
 - This model specifies the entire desired application deployment ...
 - ... and is consumed by a deployment system for fully automated execution
- Thus, this is much faster and less error-prone than the interactive way

A lot of model-based deployment technologies have been developed



They can be categorized
in *declarative* and
imperative approaches

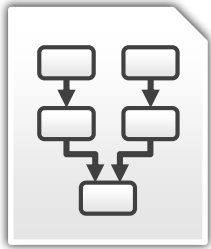


...

Declarative Deployment Model Pattern



A declarative deployment model is used to describe the desired application deployment including all components and their relationships



How to model the automated deployment of a simple application that requires only few or no individual customization?

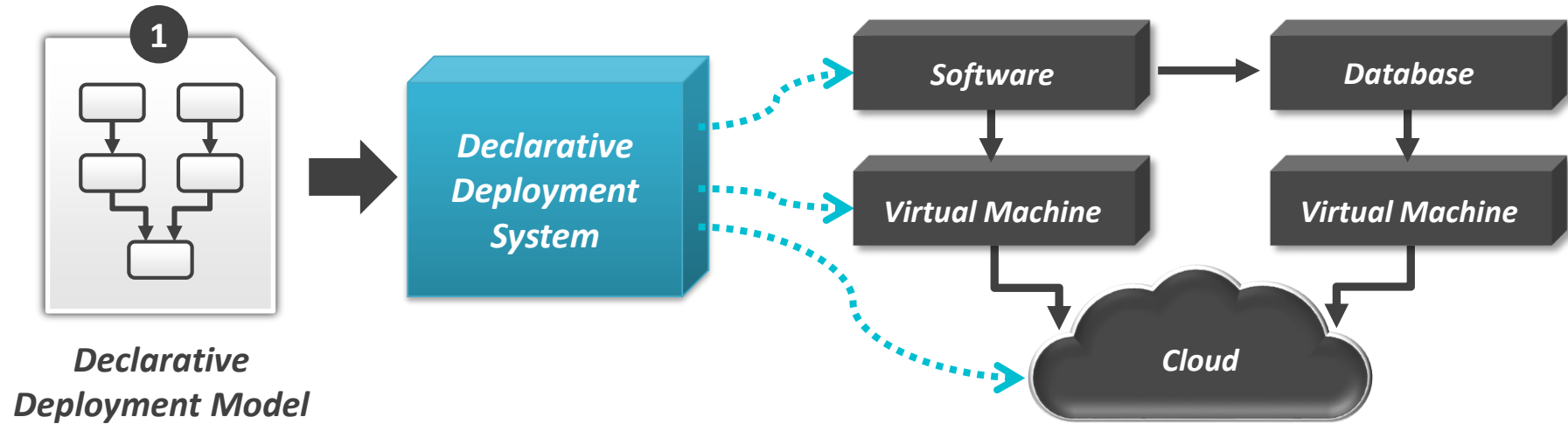
Context:

- Deployment must be automated as manual deployment is error-prone, time-consuming, and costly
- The application to be deployed is of small or medium size
- Mostly common components are used by the application, for example, a Tomcat Webserver or a MySQL database
- Only little individual deployment configuration / customization is required

Solution:

Create a declarative deployment model that defines the application's components and their relationships. Provide this model to a deployment system.

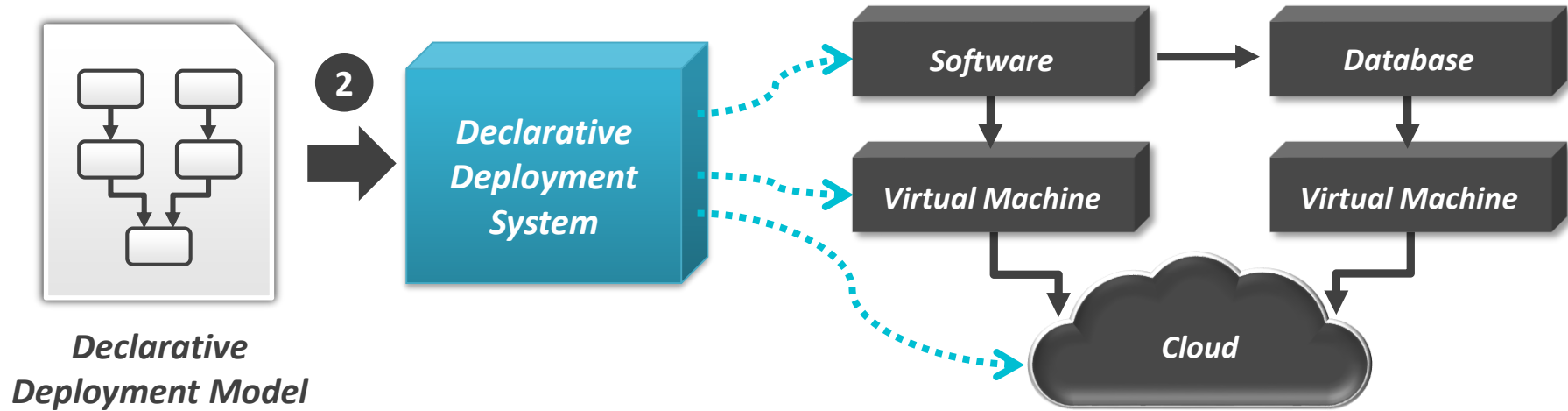
Declarative Deployment Model Pattern



Solution:

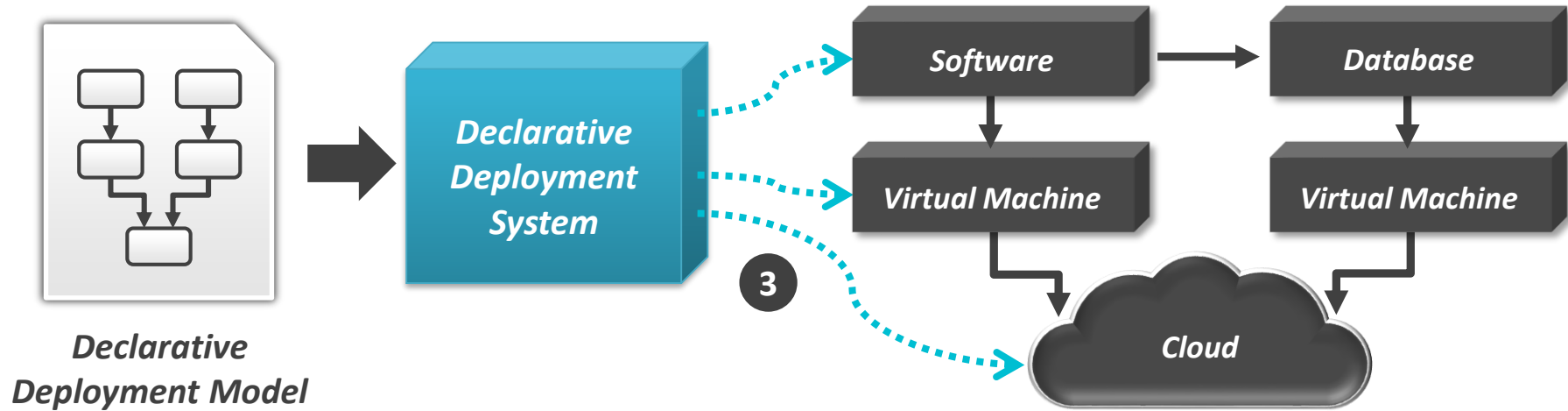
- 1 A Declarative Deployment Model describes all components to be deployed as well as their relationships
 - e.g., that a virtual machine shall be hosted on Amazon EC2 and that a certain software needs to be installed on this virtual machine
 - Type systems enable to describe the semantics of the modelled components and relationships
 - For example, to specify that the VM is of type „*Ubuntu16.04*“ and that it has a „*hostedOn*“ relationship to the component of type „*AmazonEC2*“

Declarative Deployment Model Pattern



Solution: 2 The Declarative Deployment Model is provided to a deployment system that **interprets** the model and ...

Declarative Deployment Model Pattern



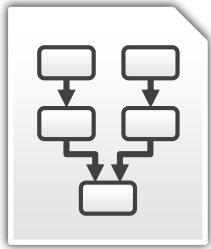
Solution:

- 3 ... executes the described deployment fully automatically
 - For example, it creates virtual machines on Amazon...
 - ... installs and starts Web Servers
 - ... deploys application files on these Web Servers
 - ... creates a MySQL database on Amazon RDS (or installs a MySQL DBMS on another virtual machine)
 - ... and connects the deployed application to this database

Declarative Deployment Model Pattern



A declarative deployment model is used to describe the desired application deployment including all components and their relationships



How to model the automated deployment of a simple application that requires only few or no individual customization?

Result:

- Declarative Deployment Models specify **what** has to be deployed ...
 - ... but not **how** the deployment has to be executed
- Automated deployment of composite applications is eased
 - Intuitive desired state modelling helps to avoid modelling errors
 - Structure of application to be deployed is reflected directly by the model
 - „What you see is what you get“

Declarative Deployment Model Pattern – Example #1

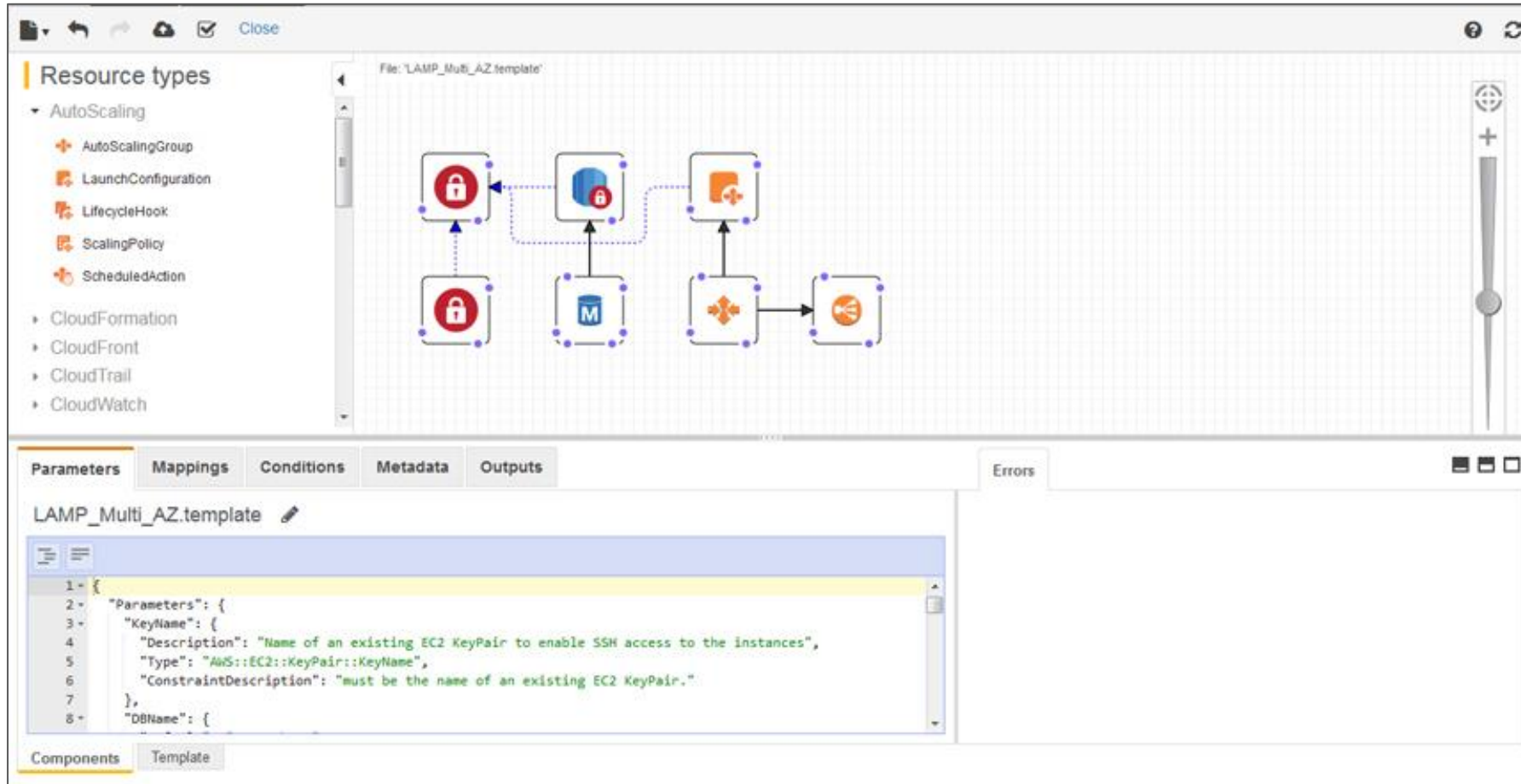


```
{
  "Description" : "Create an EC2 instance running the Amazon Linux 32 bit AMI.",
  "Parameters" : {
    "KeyPair" : {
      "Description" : "The EC2 Key Pair to allow SSH access to the instance",
      "Type" : "String"
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyPair" },
        "ImageId" : "ami-3b355a52"
      }
    }
  },
  "Outputs" : {
    "InstanceId" : {
      "Description" : "The InstanceId of the newly created EC2 instance",
      "Value" : {
        "Ref" : "Ec2Instance"
      }
    }
  }
}
```

[1]

¹ Based on and taken from <https://aws.amazon.com/cloudformation/details/> (on 05.07.2017)

Declarative Deployment Model Pattern – Example #1




[1]

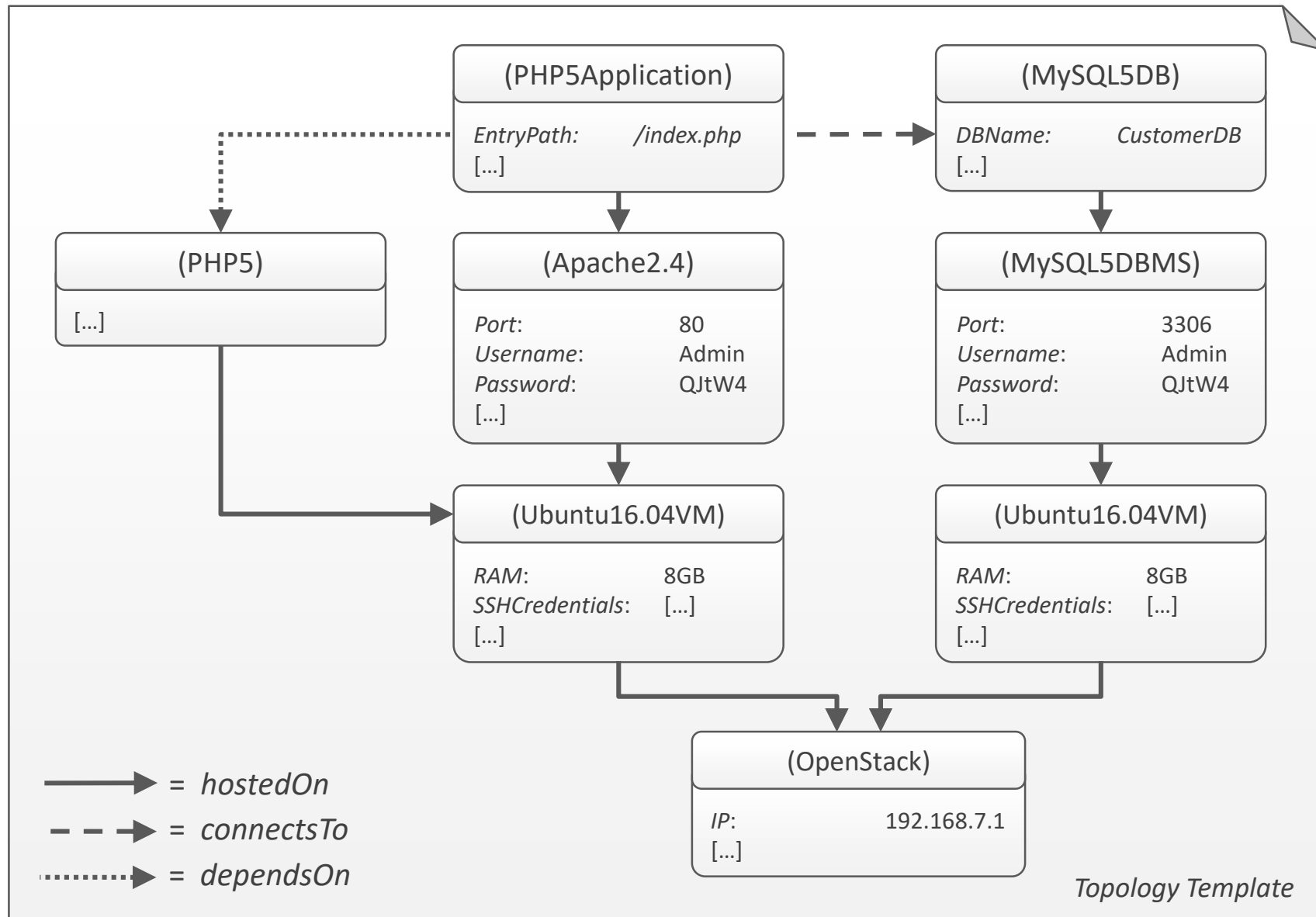
- Amazon also provides a graphical modelling tool for CloudFormation

¹ Based on and taken from <https://aws.amazon.com/cloudformation/details/> (on 05.07.2017)

Declarative Deployment Model Pattern – Example #2

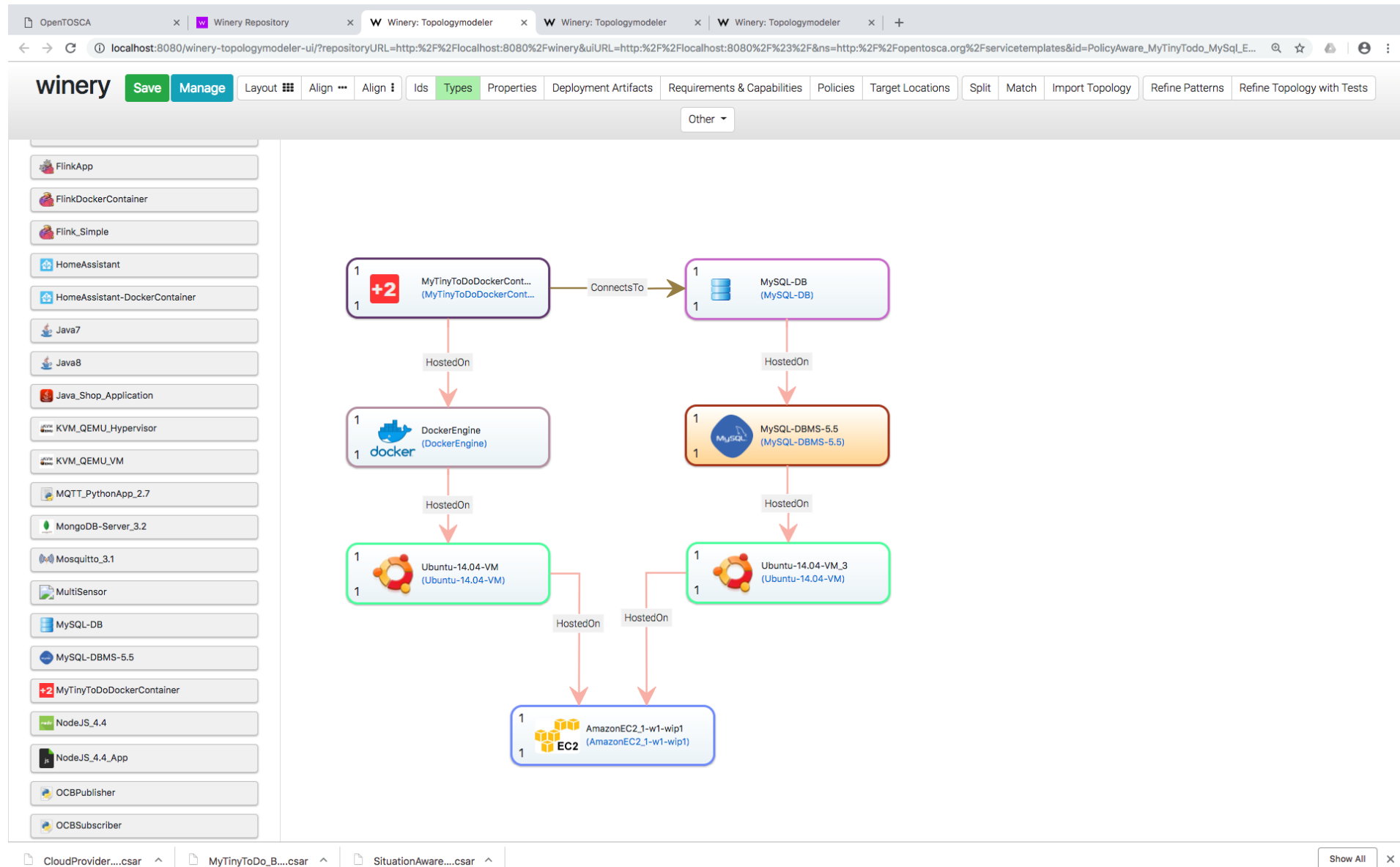
- The *Topology and Orchestration Specification for Cloud Applications (TOSCA)* is **OASIS**  standard for automating the deployment and management of cloud applications in a portable manner
- The goals of TOSCA are:
 - Automation of Deployment and Management
 - Portability
 - Interoperability
 - Vendor-neutral ecosystem
- TOSCA provides a metamodel for declarative deployment models that can be executed automatically by TOSCA runtimes

Declarative Deployment Model Pattern – Example #2



Topology Template

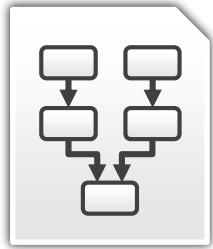
Modeling tool for TOSCA: Eclipse Winery (developed by Uni Stuttgart)



Declarative Deployment Model Pattern



A declarative deployment model is used to describe the desired application deployment including all components and their relationships



How to model the automated deployment of a simple application that requires only few or no individual customization?

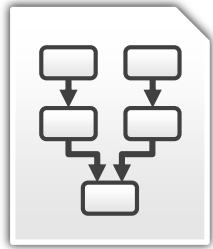
Result:

- **Less technical expertise required for creating declarative deployments models** than required for writing low-level deployment scripts
 - Only the desired application structure including all components and their relationships must be specified ...
 - ... but not technical details about the actual deployment tasks that have to be executed

Declarative Deployment Model Pattern



A declarative deployment model is used to describe the desired application deployment including all components and their relationships



How to model the automated deployment of a simple application that requires only few or no individual customization?

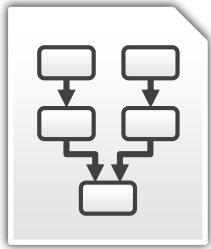
Result:

- Declarative models are **interpreted** by the deployment system that derives the actual technical deployment tasks to be executed automatically
 - The deployment process **cannot** be customized arbitrarily
 - Typically, deployment systems support that developers provide own lifecycle implementations for components
 - e.g. to provide an own *install*-implementation for a component
 - However, the customizability is typically limited to these plug-points
 - Therefore, the overall customizability of the deployment is limited

Declarative Deployment Model Pattern



A declarative deployment model is used to describe the desired application deployment including all components and their relationships



How to model the automated deployment of a simple application that requires only few or no individual customization?

Result:

- As declarative models are **interpreted** by the deployment system, an arbitrary deployment customization is **not** possible
→ **If you have a complex application, the declarative approach may fail**
- **Moreover, variations between different deployment systems are possible regarding the actually executed deployment process**
 - One system may use an SH script for installing an Apache Webserver, another system may use a configuration management technology for this

Declarative Deployment Model Pattern – Summary

- The declarative approach is intuitive and requires only little technical expertise
- However, the main drawbacks are ...
 - ... that this approach is only suited for the deployment of standard applications that consist of well-known, common components
 - ... that we cannot customize the deployment process arbitrarily
- **So the question is:** *“Why not specifying the deployment process by ourselves?”*
 - For example, by implementing a Java program, SH script, or BPEL / BPMN workflow that executes all required technical deployment tasks
 - We could implement exactly what we want to have
 - We could customize the deployment process arbitrarily
 - The deployment process would not vary between different deployment systems

→ This is the imperative approach, see next slide 😊

Imperative Deployment Model Pattern



An imperative deployment model is used to explicitly describe all technical deployment tasks to be executed for deploying an application



How to model the automated deployment of a complex application that requires application-specific customization?

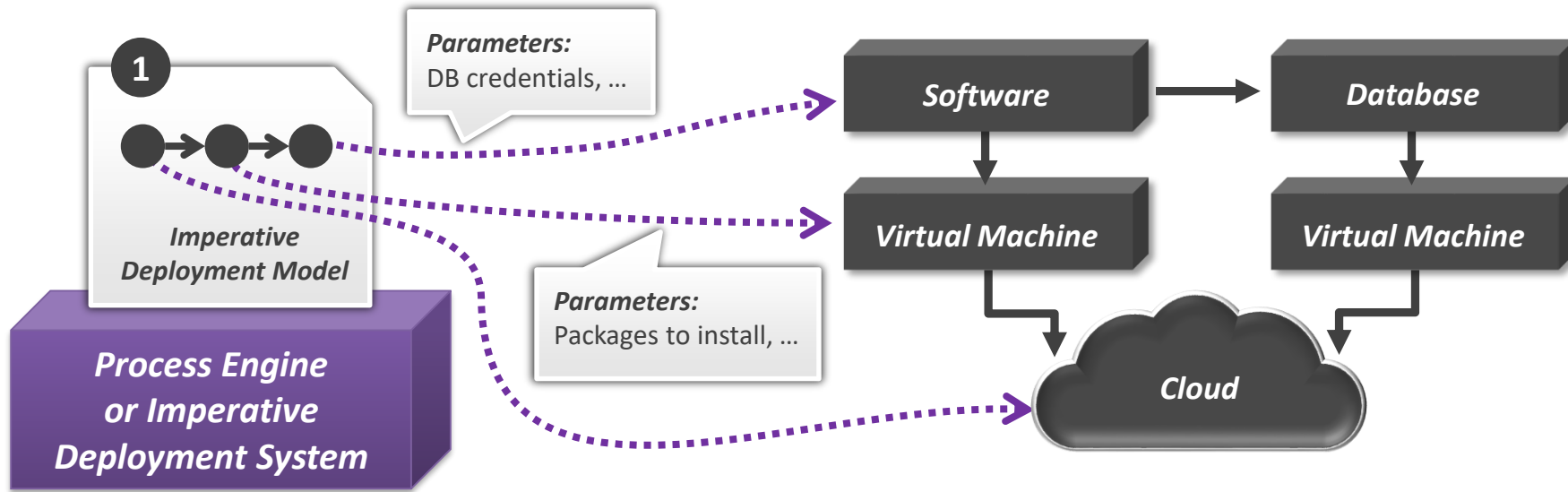
Context:

- Manual deployment is error-prone, time-consuming, and costly
- The application to be deployed is **large** or **complex**
- Besides common components also **custom components** are used by the application, for example, a proprietary CMS
- **Application-specific deployment configuration / customization is required**
- The deployment process must **not** vary

Solution:

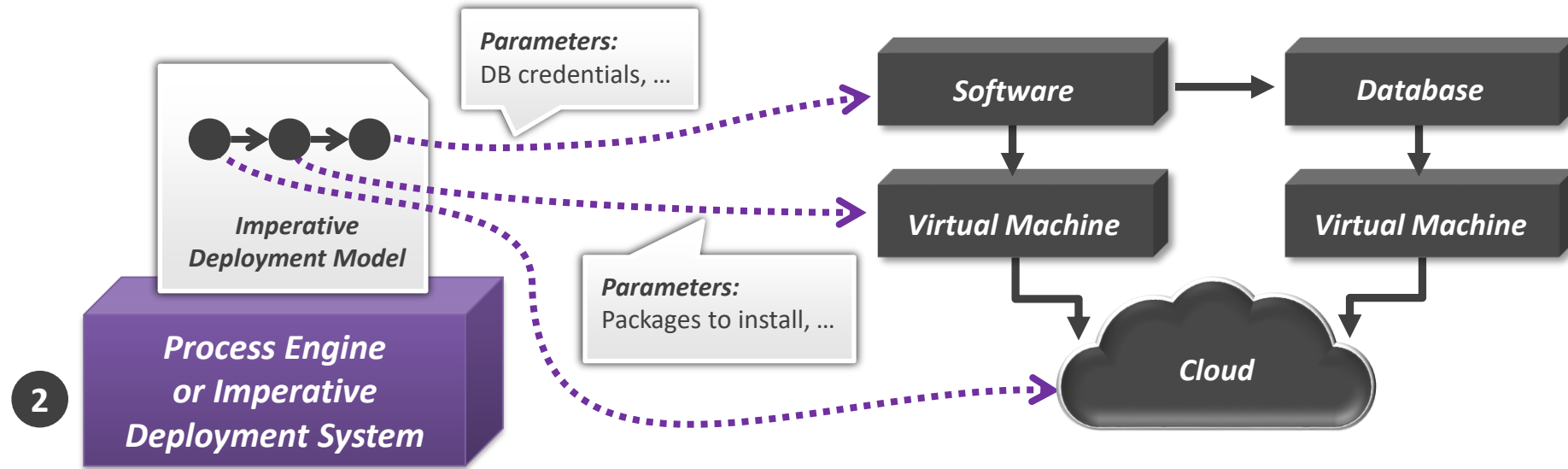
Create an imperative deployment model that explicitly specifies all technical deployment tasks to be executed as well as the control- and data-flow. Provide the model to a process execution engine or to an imperative deployment system.

Imperative Deployment Model Pattern



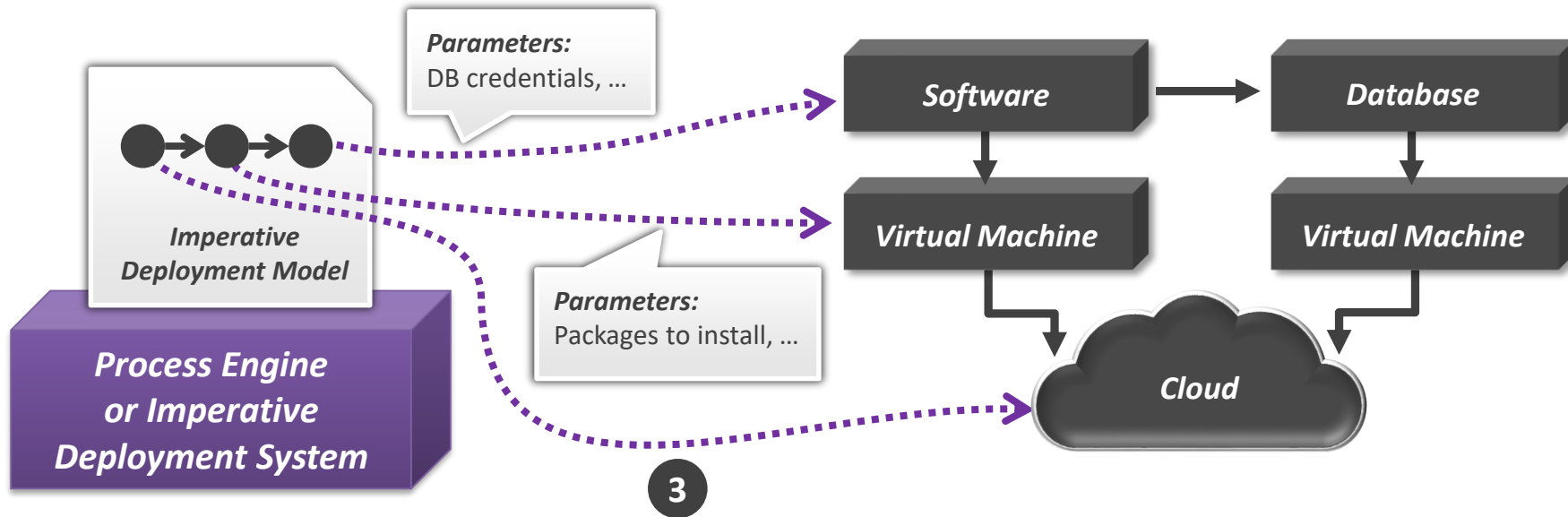
- Solution:**
- 1 An Imperative Deployment Model describes all *technical deployment tasks* to be executed in the form of an *process*
 - For example, a deployment task can be an API invocation, a script execution, or copying files via SSH and SCP onto a virtual machine
 - Thus, Imperative Deployment Models are *process models* that also specify the order of the tasks as well as the data flow between them
 - Imperative Deployment models can be realized using well-known process execution technologies such as BPEL or BPMN workflows

Imperative Deployment Model Pattern



- Solution:**
- 2 The imperative deployment model is used by a *process engine* or an *imperative deployment system* that executes the process model as specified
 - This execution is unique due to defined operational semantics

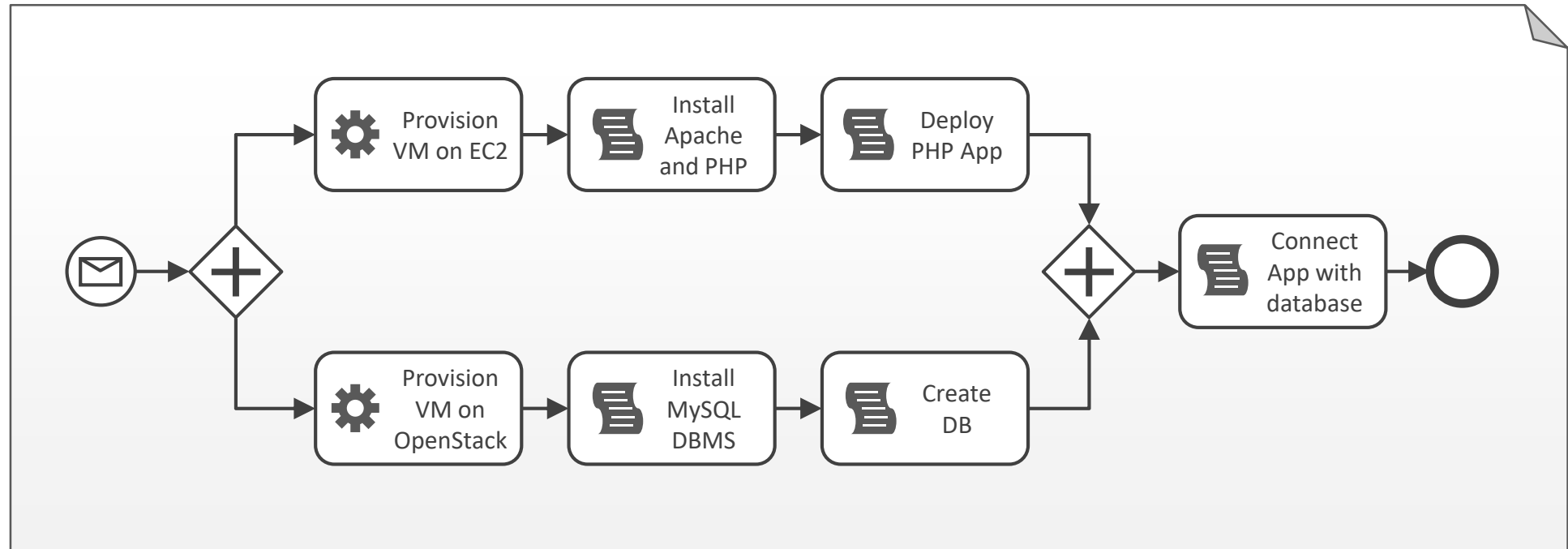
Imperative Deployment Model Pattern



Solution:

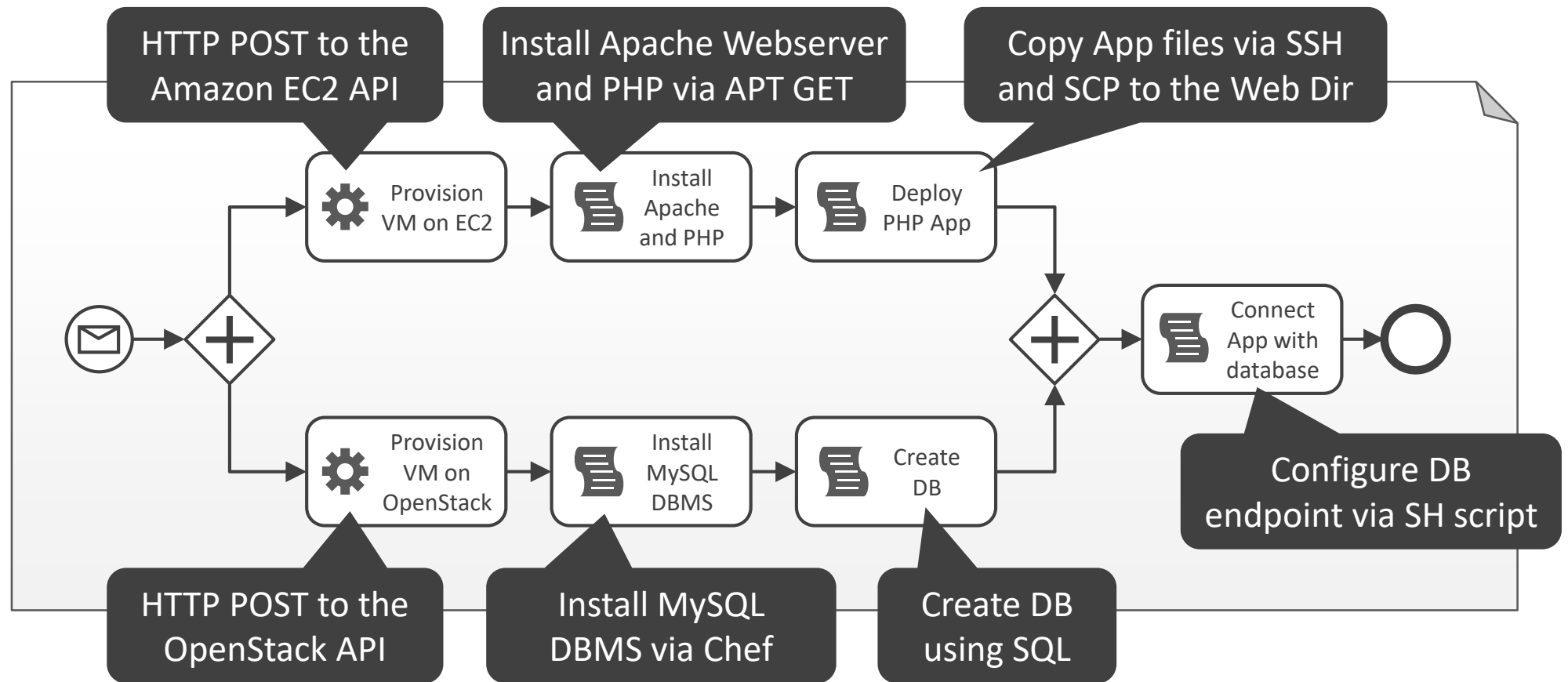
- 3 The execution of the imperative deployment model results in various calls to APIs, executions of SH scripts on virtual machines, etc.
 - Each task must be specified in every technical detail:
 - Parameters for API calls
 - Parameters for script executions
 - SSH credentials for copying and executing a script on a virtual machine
 - etc.

Imperative Deployment Model Pattern - Example



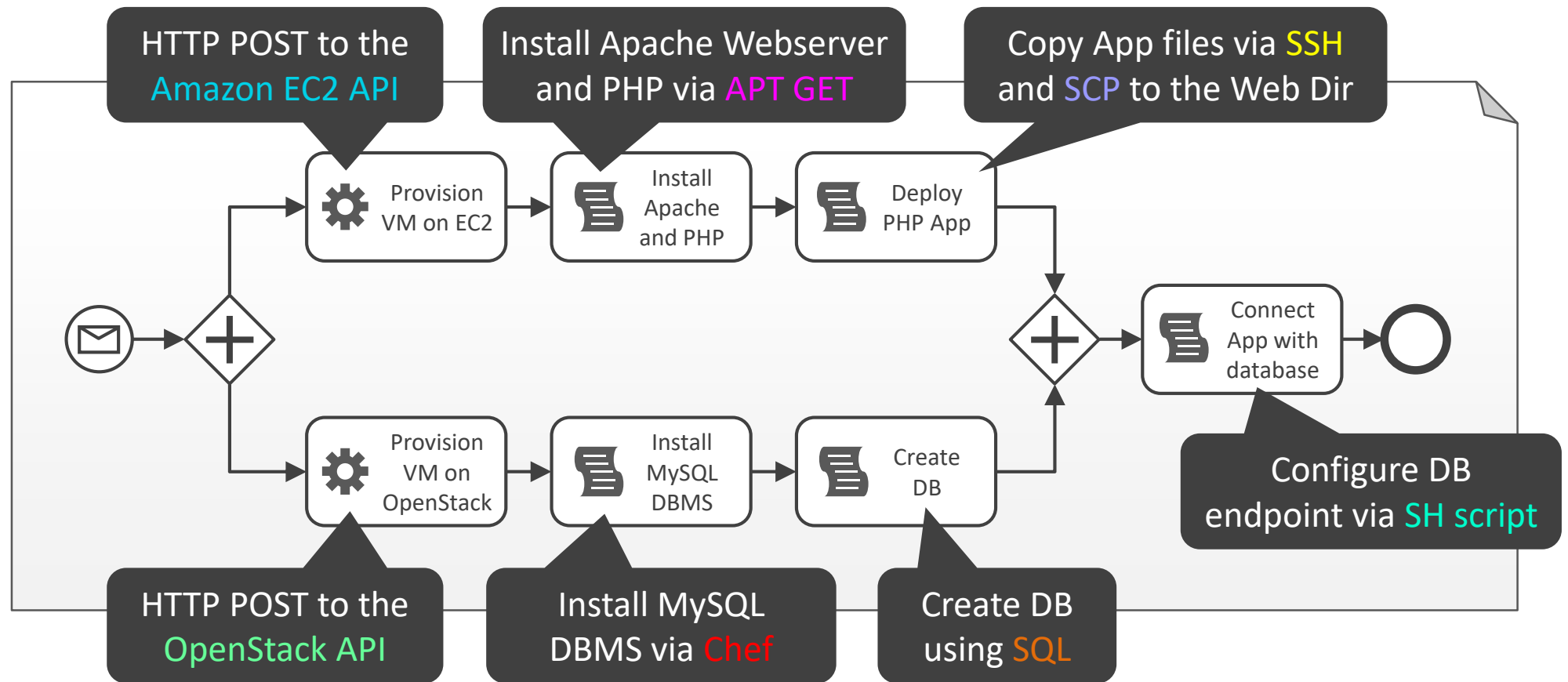
- BPMN process model for the automated provisioning of a hybrid cloud application
 - Service Tasks and Script Tasks execute the API calls and installation operations
 - **Please note:** To recognize what will be deployed, we have to “read” the process
→ In contrast, a declarative deployment model shows the result directly

Imperative Deployment Model Pattern - Example



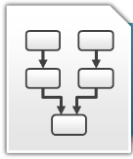
- Service Tasks and Script Tasks execute the API calls and installation operations
 - Following the order specified by the control flow
 - We omit the data flow for simplicity

Imperative Deployment Model Pattern - Example



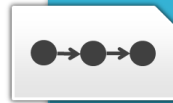
- Each **color** represents a different technology
 - For creating this process model, **immense technical expertise is required**
 - **But on the other side, this enables a full customization of the provisioning**

Comparison declarative vs. imperative deployment modelling



Declarative Deployment Model

- + Intuitive and fast desired application state modelling
- + Deployment model intuitively reflects the system's structure
- + Simplifies common and noncomplex application deployments
- + Requires less technical deployment expertise than the imperative approach
- No arbitrary deployment customization possible regarding the executed tasks
- State-preserving tasks cannot be described or only in a limited manner
- The actually executed deployment process may vary between different systems



Imperative Deployment Model

- + Arbitrary deployment logic can be modelled
- + Supports state-preserving and state-changing tasks

A **state-changing task** changes the state of one or more components or relationships of the application

For example, stopping a Webserver

A **state-preserving task** does not change the state of one or more components or relationships of the application

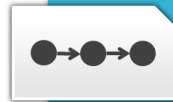
For example, export data from a DB

Comparison declarative vs. imperative deployment modelling



Declarative Deployment Model

- + Intuitive and fast desired application state modelling
- + Deployment model intuitively reflects the system's structure
- + Simplifies common and noncomplex application deployments
- + Requires less technical deployment expertise than the imperative approach
- No arbitrary deployment customization possible regarding the executed tasks
- State-preserving tasks cannot be described or only in a limited manner
- The actually executed deployment process may vary between different systems



Imperative Deployment Model

- + Arbitrary deployment logic can be modelled
- + Supports state-preserving and state-changing tasks
- + Suited for custom and complex application deployments
- + Due to well-defined operational semantics, the deployment process does not vary
- Requires immense technical deployment expertise
- Modelling is complex, error-prone, and time-consuming
- If application architecture changes, process model needs to be adapted

Major questions are...

- Which deployment modeling style is appropriate / required to automate the deployment of a certain distributed system
- Can this decision be automated?

Available deployment technologies

- There are several technologies available that support the declarative approach or the imperative approach (or both)
 - Cloud provider DSLs and APIs
 - Amazon CloudFormation, Amazon AWS API, Microsoft Azure API, ...
 - Cloud abstraction layers that also contain deployment systems
 - OpenStack, DeltaCloud, ...
 - Proprietary deployment systems
 - IBM, HP, ...
 - Script-based configuration management technologies
 - Chef, Puppet, Juju, shell scripting, ...
 - etc ...

Problems of available technologies

- However, each deployment technology employs its own...
 - ... API(s)
 - ... domain-specific language(s) (DSLs)
 - ... invocation mechanisms
 - ... data model
 - ... wording
 - ... fault handling
 - ... security mechanisms
- Moreover, many technologies are coupled to a certain provider
 - For example, CloudFormation cannot be used to deploy parts of a multi cloud application on Amazon and other parts on Google's Cloud

Problems of available technologies

- These issues result in the following three main problems

1. **Selecting a deployment system leads to a lock-in**

- If we want to replace the deployment system, this typically requires big effort as all deployment models must be adapted
- Moreover, we have to acquire expertise and educate our staff on the new system

2. **Deployment systems differ in their functionalities**

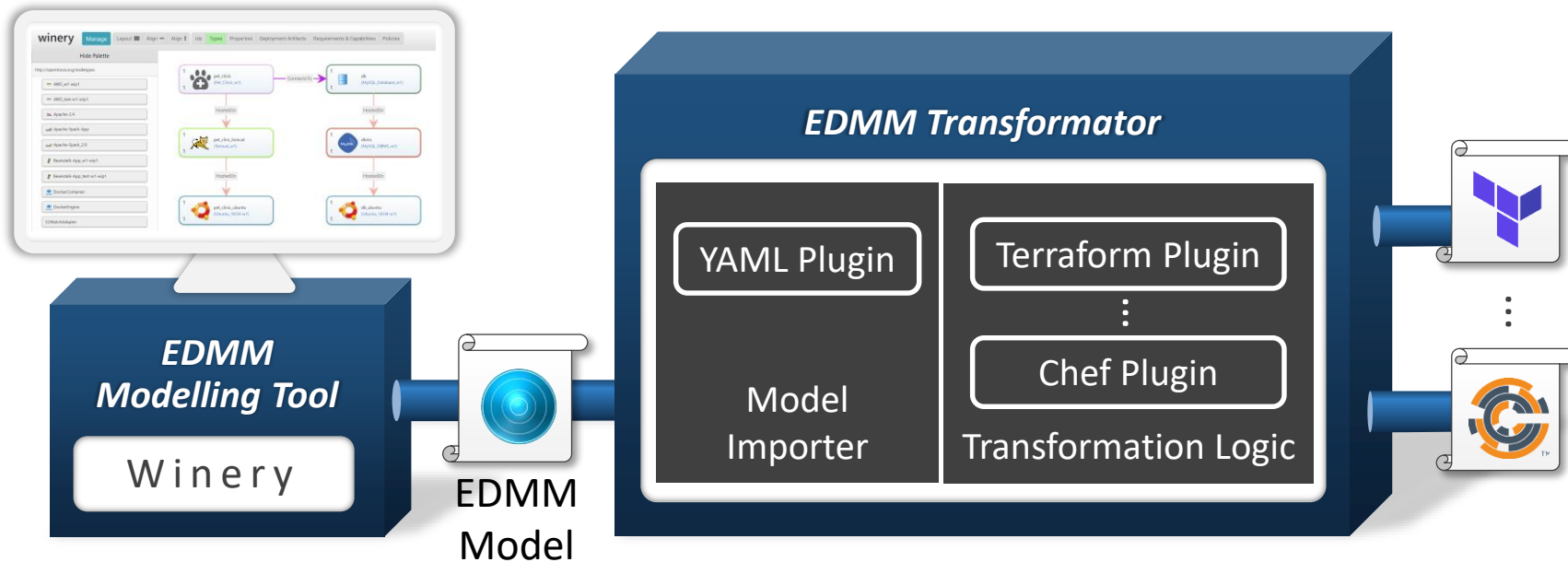
- Selecting the “best system” is almost impossible as you cannot foresee the functionalities you need in the future

3. **More and more applications need to be deployed as multi-cloud applications**

- Only a few systems support this
→ Often multiple technologies have to be combined

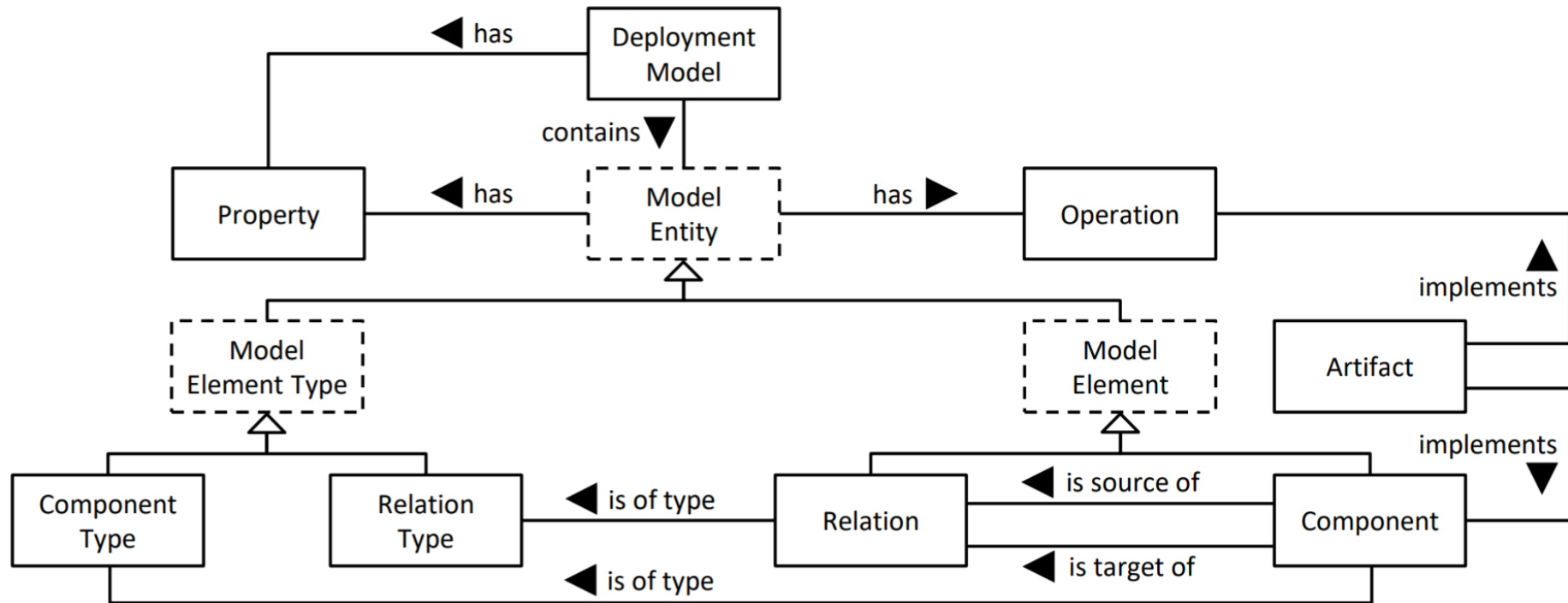
The Essential Deployment Metamodel (EDMM)

The Essential Deployment Metamodel (EDMM)

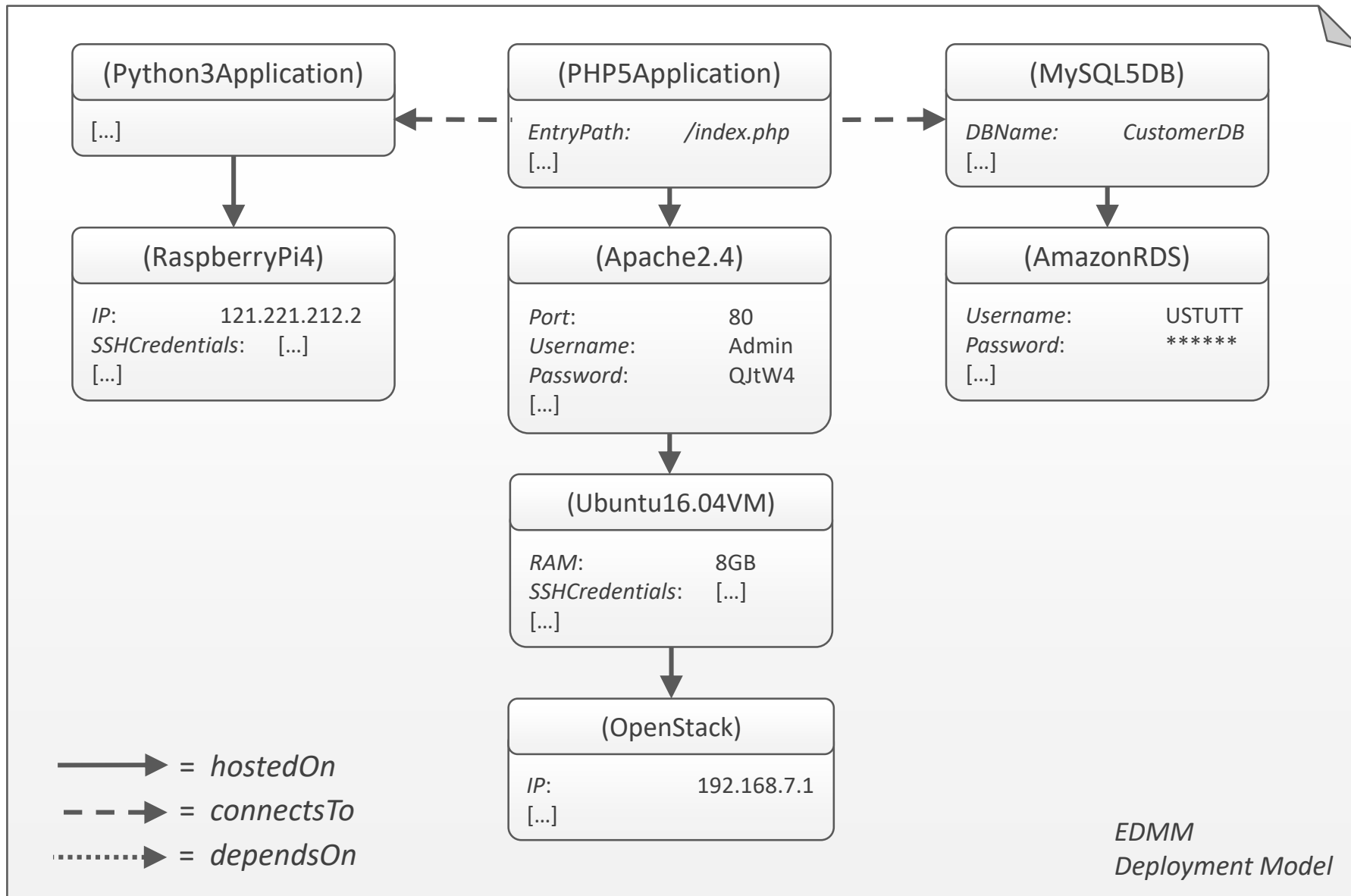


[1] Wurster, M., et al.: The Essential Deployment Metamodel: A Systematic Review of Deployment Automation Technologies. SICS Software-Intensive Cyber-Physical Systems (Aug 2019)

Key features every deployment technology supports



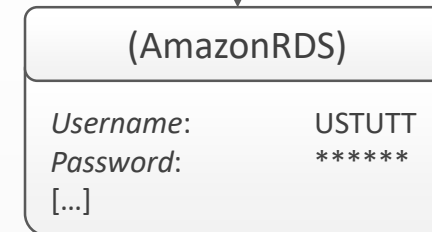
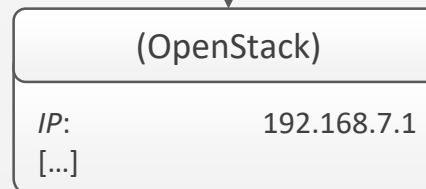
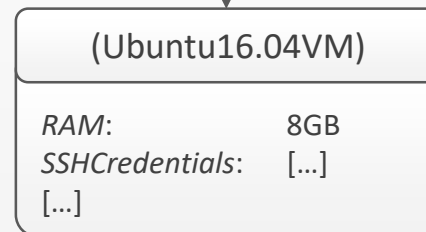
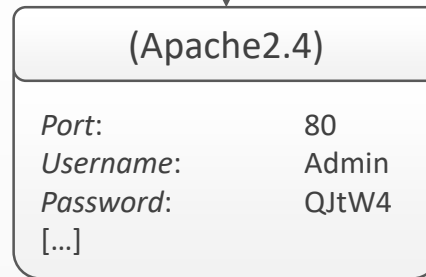
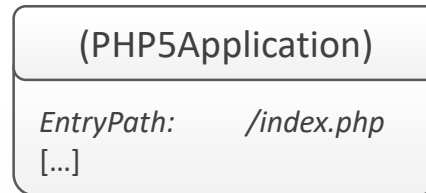
Current Research: Deployment Technology Orchestration



Current Research: Deployment Technology Orchestration

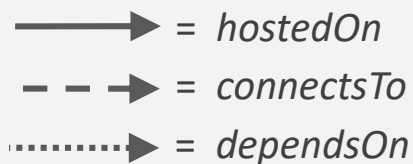
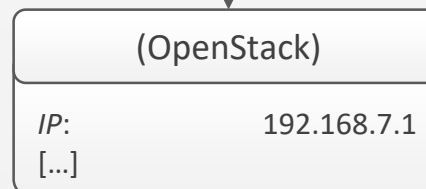
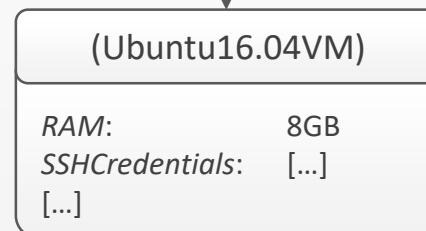
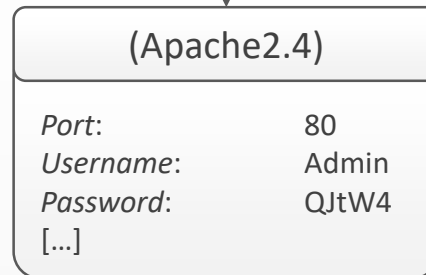
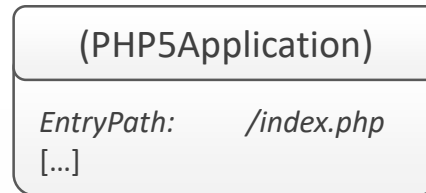


—→ = *hostedOn*
--→ = *connectsTo*
.....→ = *dependsOn*



EDMM
Deployment Model

Current Research: Deployment Technology Orchestration



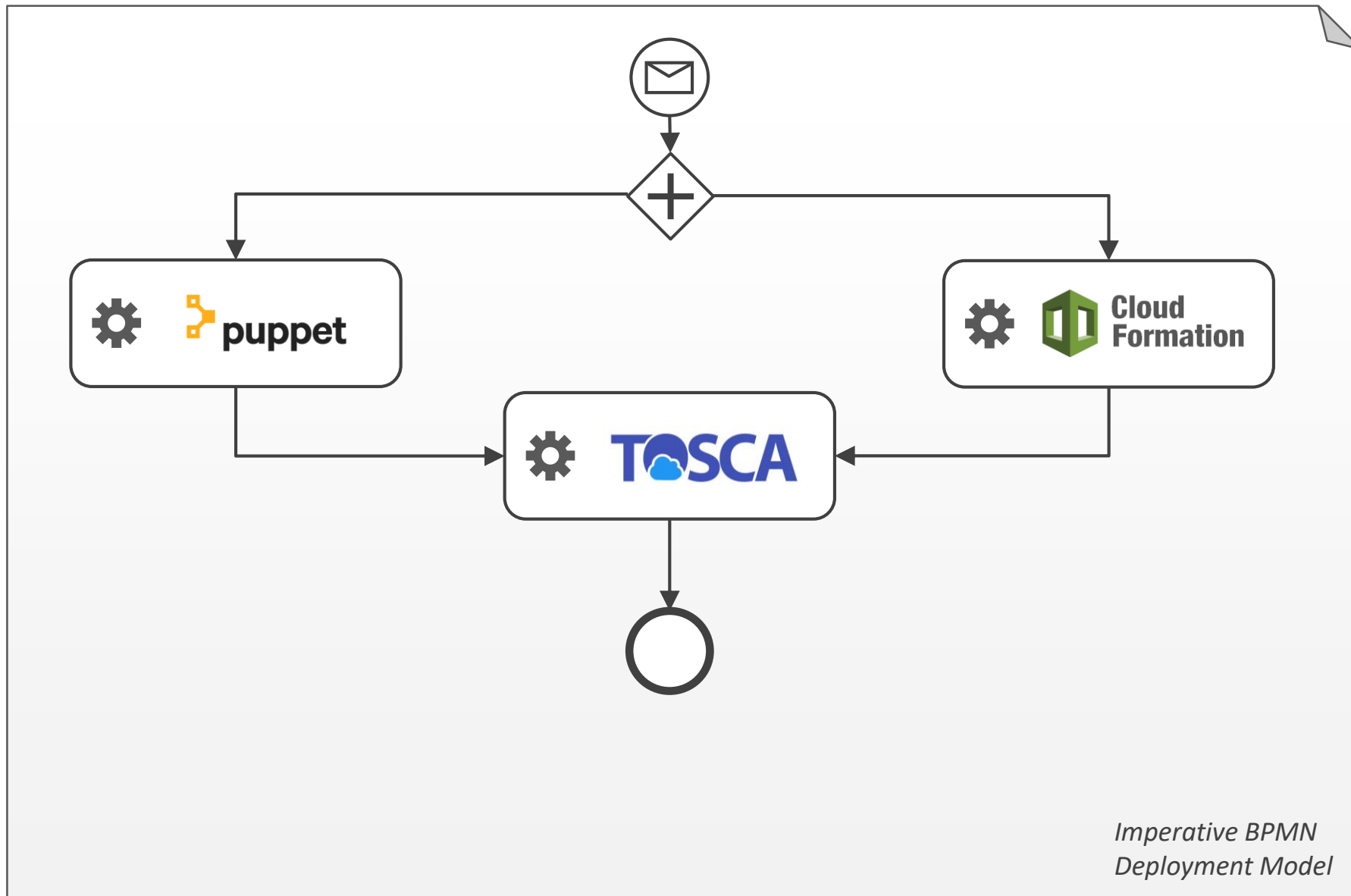
EDMM
Deployment Model

Current Research: Deployment Technology Orchestration

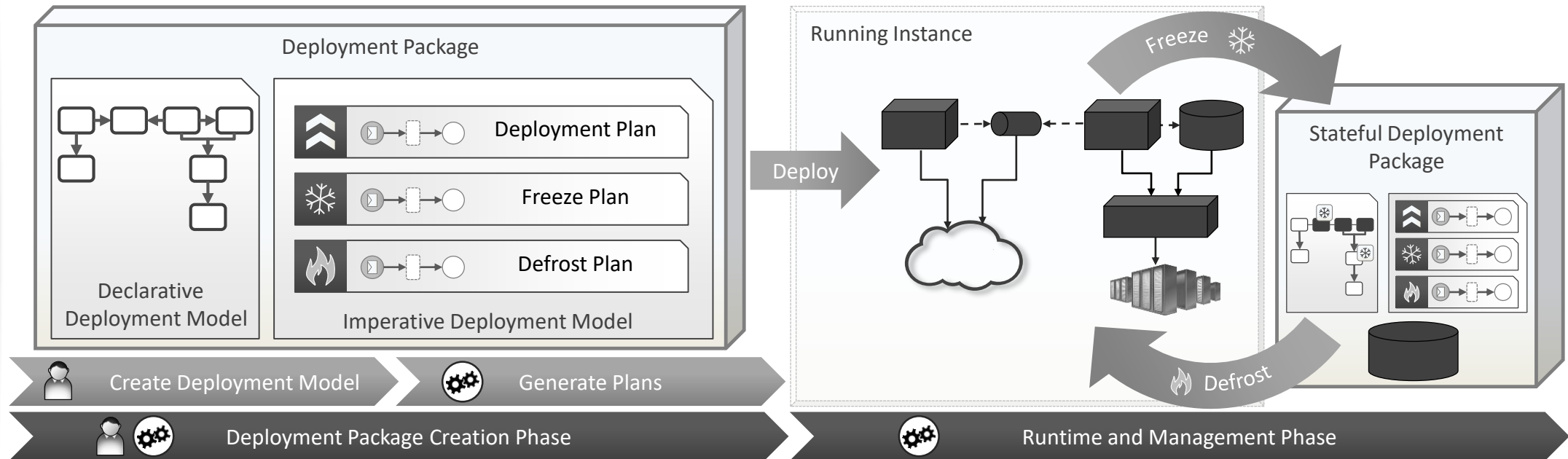


EDMM
Deployment Model

Current Research: Deployment Technology Orchestration



Freezing and Defrosting Cloud Applications – Overview



Thank you very much 😊