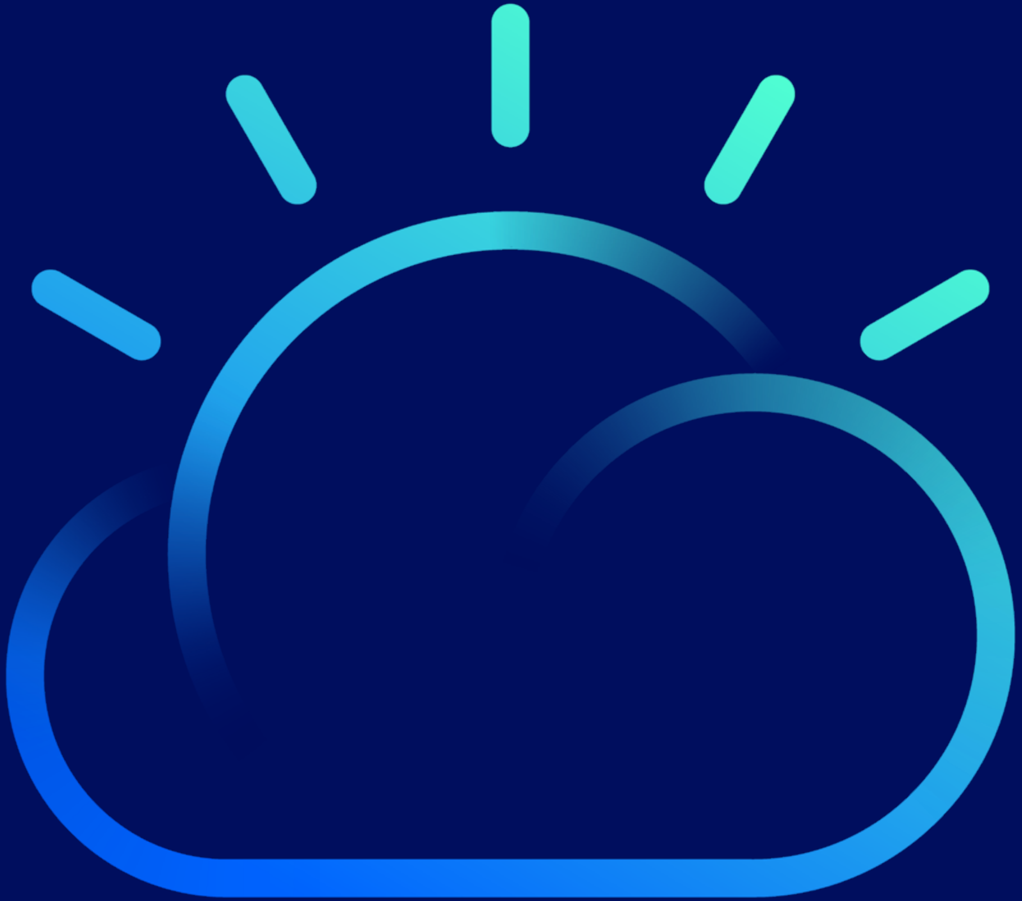
IBM Cloud Automation Manager Bootcamp

Introduction to CAM



IBM Cloud

Overview

What is CAM?

Choose the cloud implementation that works best **for you.**

Centralized Hybrid connectivity, management & DevSecOps

Next-gen middleware, integration data analytics & science

Open Source Platform

IBM **Cloud**

Next-gen middleware, integration, data analytics & science

Applications, solutions & services

Blockchain, financial services, IoT, Mobile, etc.

Cloud Infrastructure

A highly scalable, security-enabled infrastructure

AI

Cognitive building blocks for developers

IBM **Cloud** Automation Manager (multi-cloud)

Automate and standardize delivery of infrastructure and application environments in multiple clouds

Compose complex application environments into services that are easy to consume

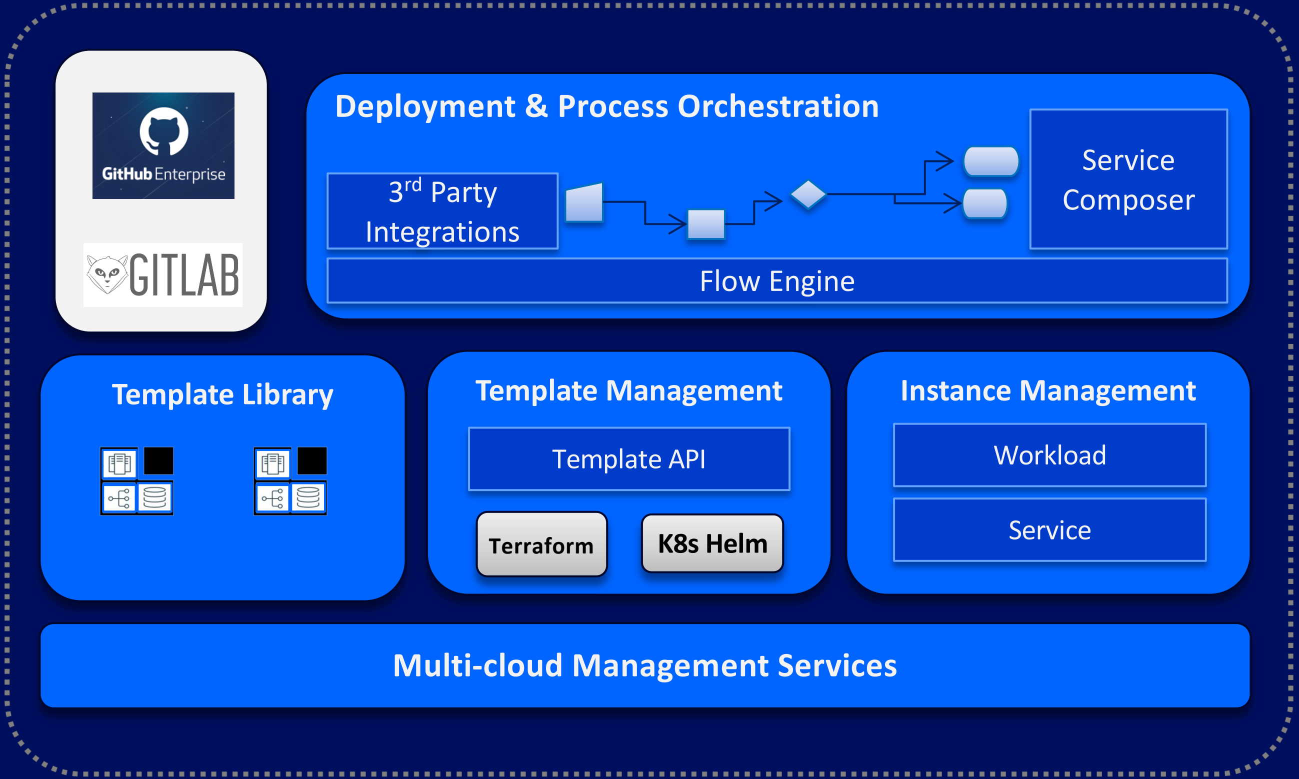
Manage workload and service lifecycle across multiple clouds

Introducing Cloud Automation Manager: *Use cases*

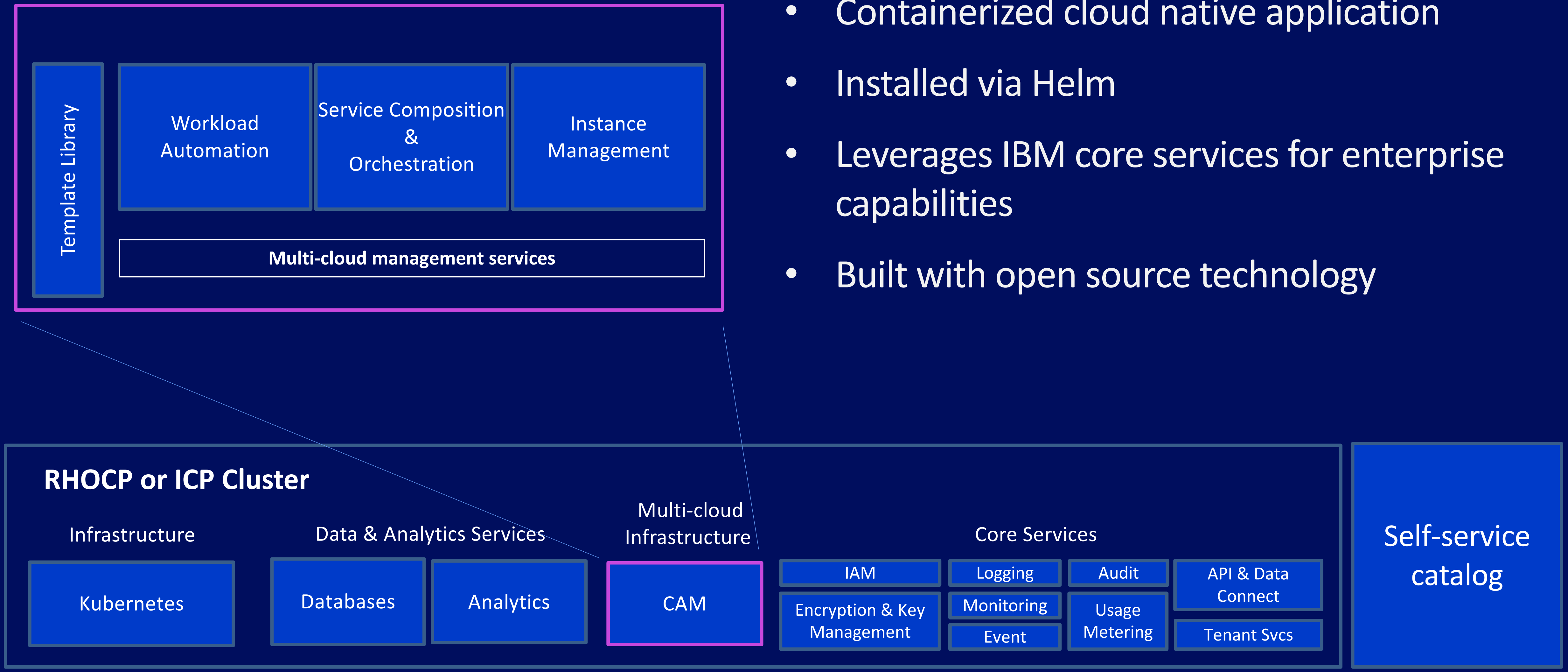
- ***Automate and standardize*** delivery of *infrastructure and application stacks* consistently in many clouds

- ***Compose and orchestrate*** *complex environments* into *easy to consume cloud services* that can be *accessed with a DevOps tool chain* or *published into a self-service catalog*

- ***Manage workload and service instance*** *lifecycle across many clouds*



Cloud Automation Manager



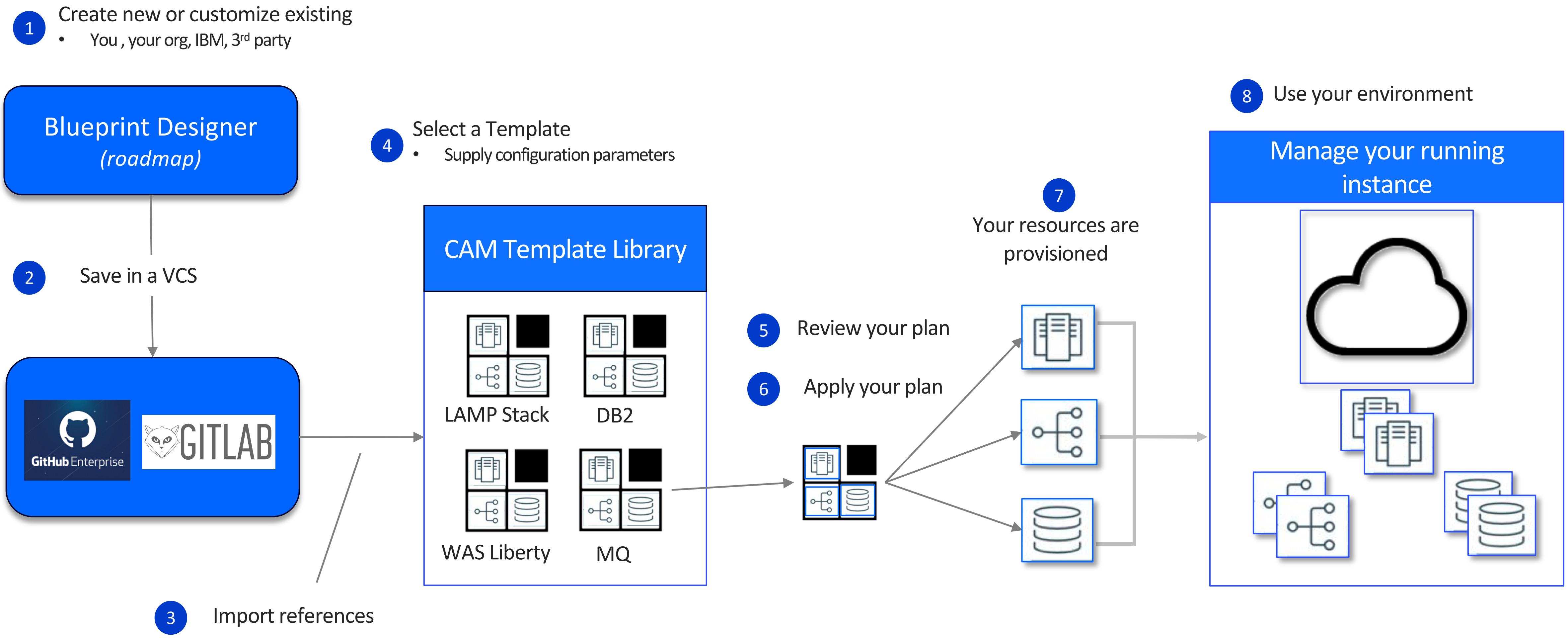
- Containerized cloud native application
- Installed via Helm
- Leverages IBM core services for enterprise capabilities
- Built with open source technology

Automating Infrastructure with Terraform

Terraform codifies cloud APIs into a declarative text language used to describe your cloud resources

- ✓ Describe cloud state in a text based configuration file
  - *Store in a version control system*
- ✓ ***Plan*** the configuration to do a test dry run
  - *The plan tells you what will be destroyed, added, or modified*
- ✓ ***Apply*** the configuration to make the configuration real
  - *The apply causes the Terraform engine to invoke cloud provider APIs*
- ✓ Use ***plan*** *and* ***apply*** to update running infrastructure
  - *Modify configuration file (new version, cpu, memory, number of VMs, etc)*
  - *Manually ‘taint’ selected resources to trigger full restart*
  - ***Plan*** *to validate then* ***Apply*** *to make the change real*

Automation with Terraform
is simple and easy



1 Create new or customize existing
- You , your org, IBM, 3rd party

Blueprint Designer *(roadmap)*

2 Save in a VCS

3 Import references

4 Select a Template
- Supply configuration parameters

CAM Template Library

5 Review your plan

6 Apply your plan

7 Your resources are provisioned

8 Use your environment

Manage your running instance

Infrastructure as Code (IAC)

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools [1]

✓ Programmable & scriptable

- *Manage infrastructure with DevOps practices*
- *REST API*

✓ Reproducible

- *Sharable and on-demand*
- *Repeatable processes*

✓ Improve governance and transparency

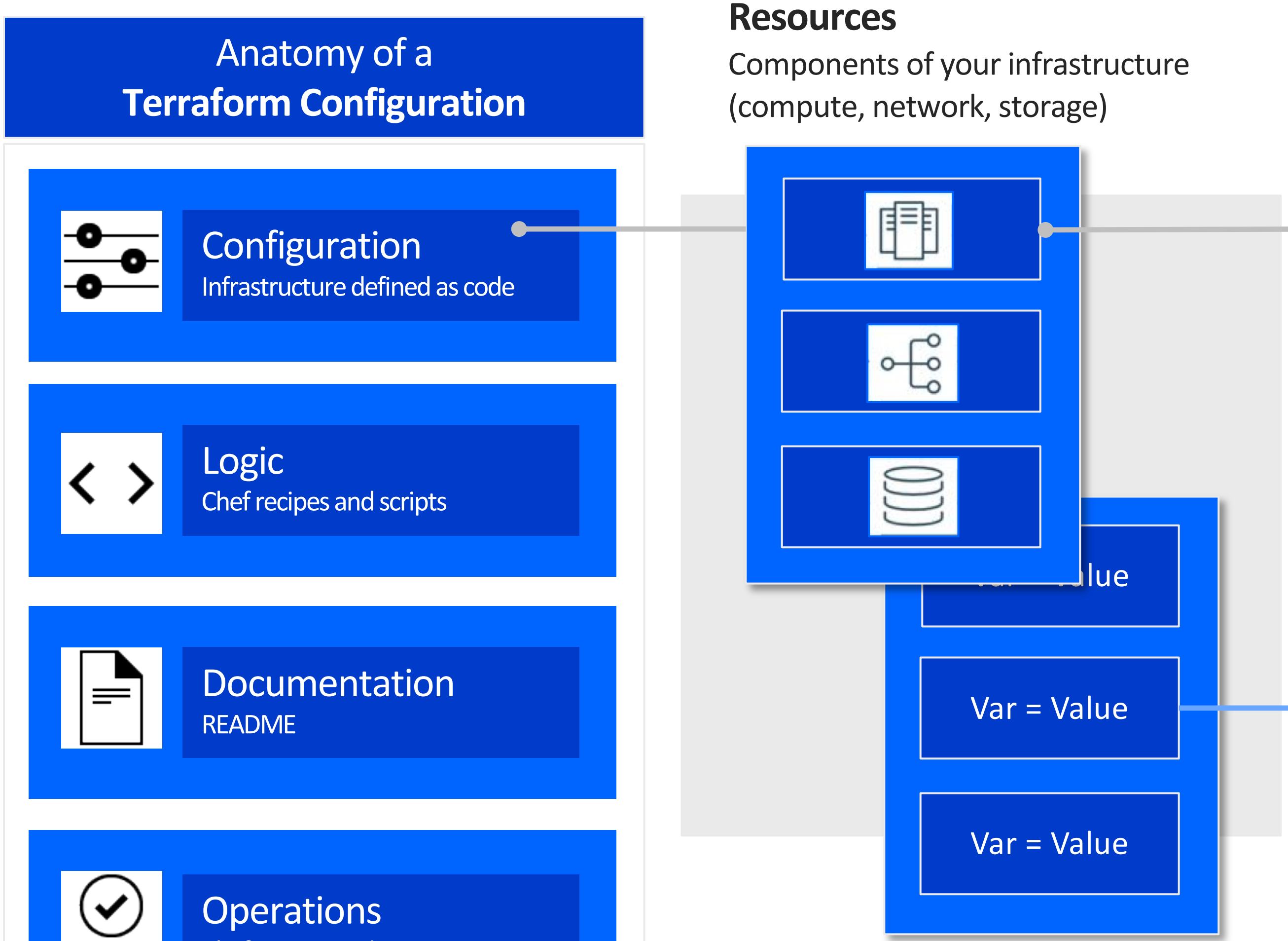
- *Track the ‘who’, ‘what’, ‘when’ and ‘why’ of all environment changes*

✓ Supports the ***Immutable Infrastructure*** design style

- *Disposable infrastructure*
- *Redeploy rather than patch*

[1] Source: https://en.wikipedia.org/wiki/Infrastructure_as_Code

Anatomy of a Terraform Configuration



Anatomy of a
Terraform Configuration

Configuration
Infrastructure defined as code

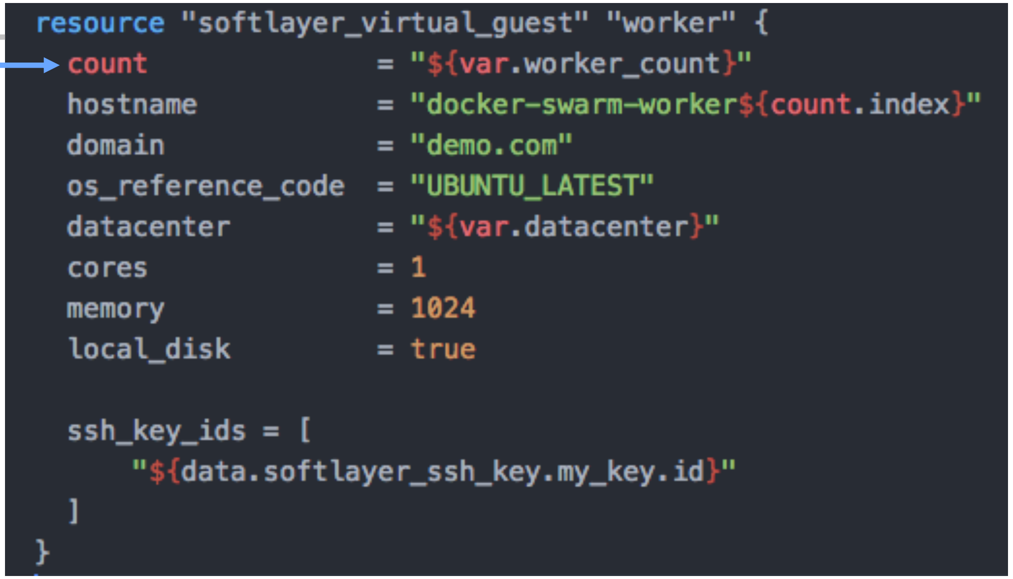
Logic
Chef recipes and scripts

Documentation
README

Operations
Chef recipes and scripts

Resources
Components of your infrastructure (compute, network, storage)

Resource Configuration
For a component in your infrastructure



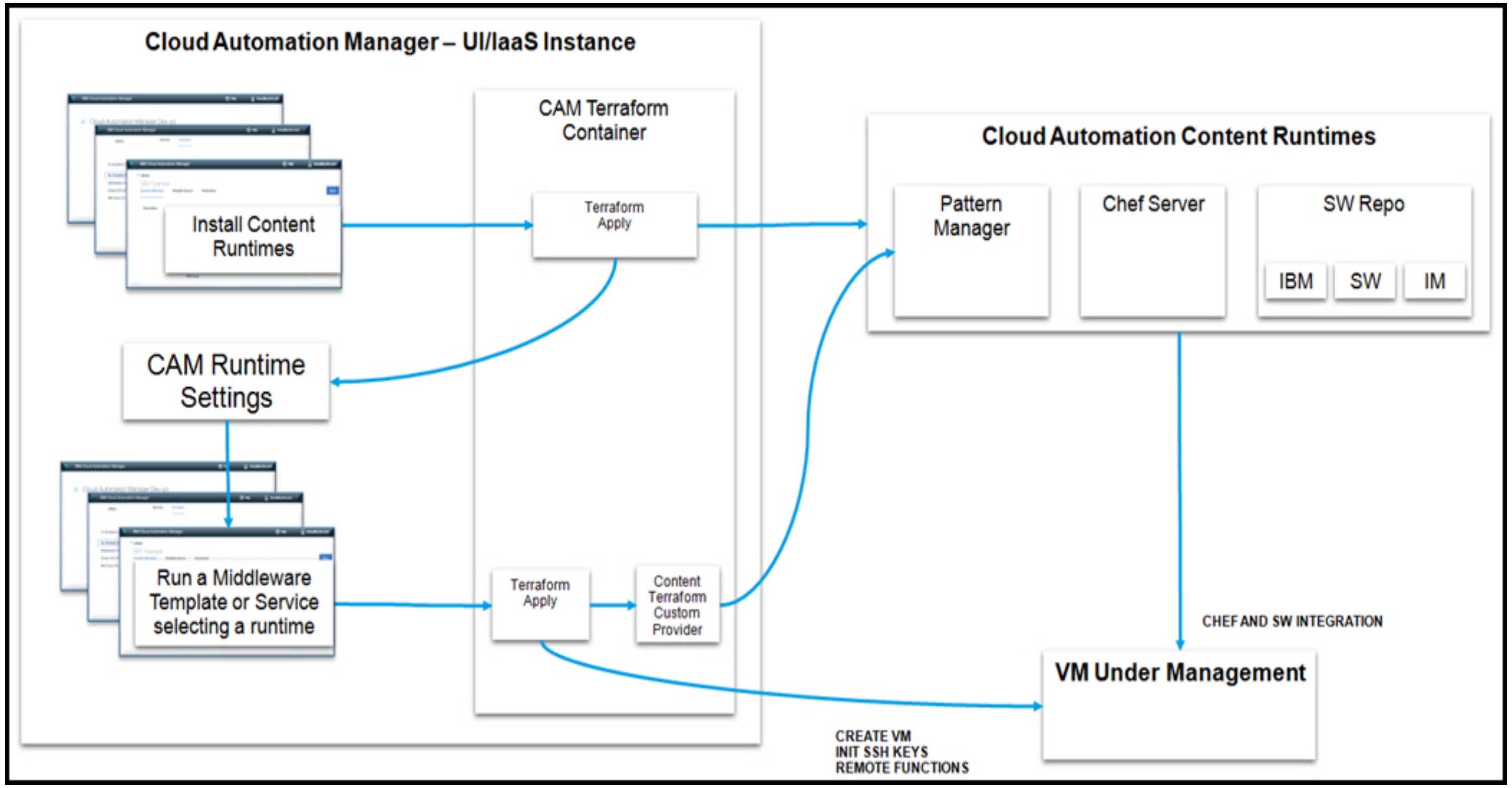
```
resource "softlayer_virtual_guest" "worker" {
  count              = "${var.worker_count}"
  hostname           = "docker-swarm-worker${count.index}"
  domain             = "demo.com"
  os_reference_code  = "UBUNTU_LATEST"
  datacenter         = "${var.datacenter}"
  cores              = 1
  memory             = 1024
  local_disk         = true

  ssh_key_ids = [
      "${data.softlayer_ssh_key.my_key.id}"
  ]
}
```

Variables
Define the parameters of your resource

Cloud Automation Manager Chef Content Runtime Support

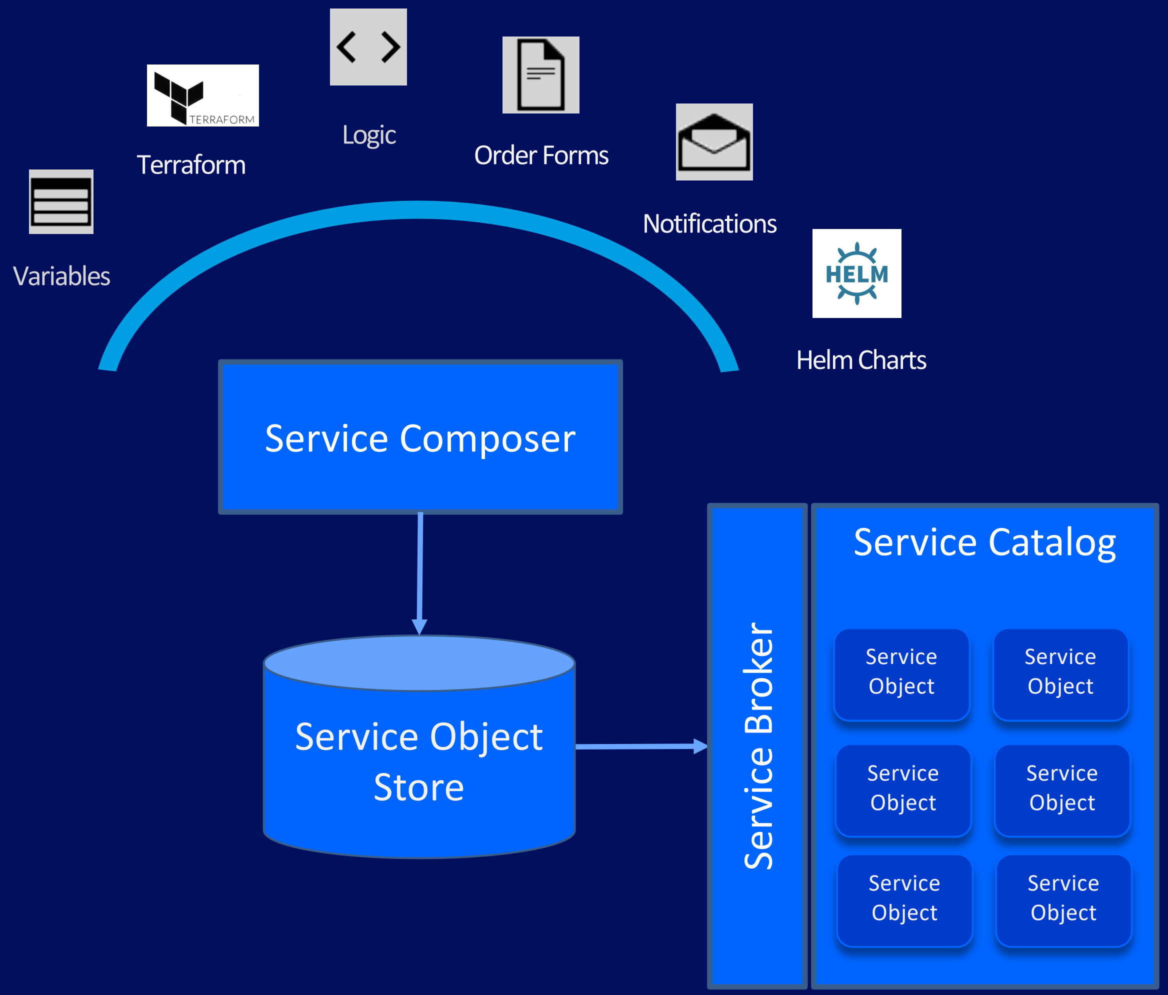
Built-in Chef Server runtime



- Optional Chef runtime that can be easily deployed into your provider cloud to support IBM Chef enabled content (VMware only)
- You supply the virtual server, let CAM stand-up a pre-configured Chef runtime
- Fully containerized
- Integrated software repository

DevOps ready ***application environments as a service*** to improve developer velocity

- ***Graphically compose services*** *to hide automation complexity, lock down configurations and simplify end-user consumption*
- ***Publish composed services*** *to self service catalogs*
- *Consume services via REST API*



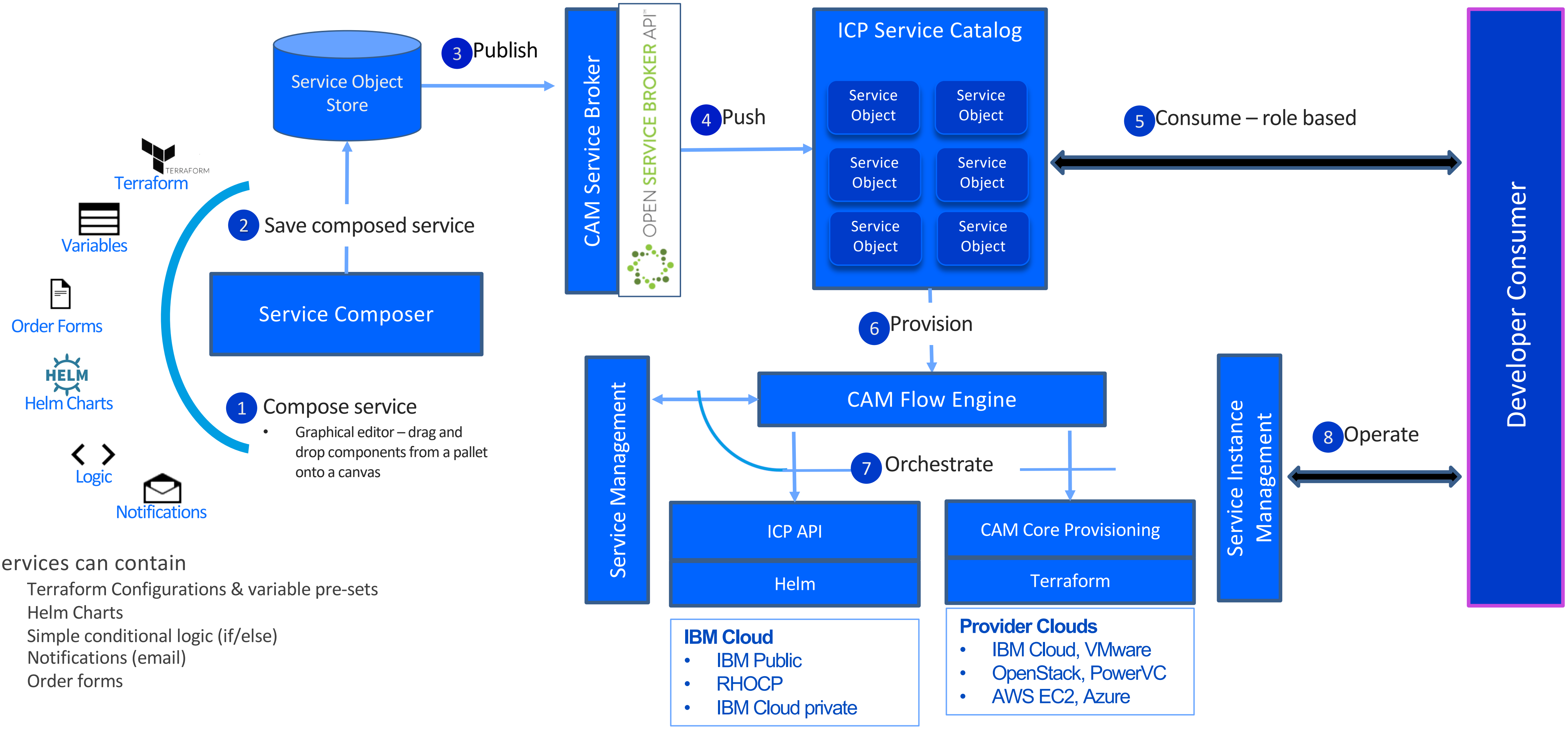
Service Management Lifecycle:
Compose, Publish, Consume and Operate

Services are published into a service catalog

- Leveraging Open Service Broker API

Roadmap

- Support more advanced workflows
- IBM BPM integration



3 Publish

4 Push

5 Consume – role based

6 Provision

7 Orchestrate

8 Operate

2 Save composed service

1 Compose service

- Graphical editor – drag and drop components from a pallet onto a canvas

IBM Cloud

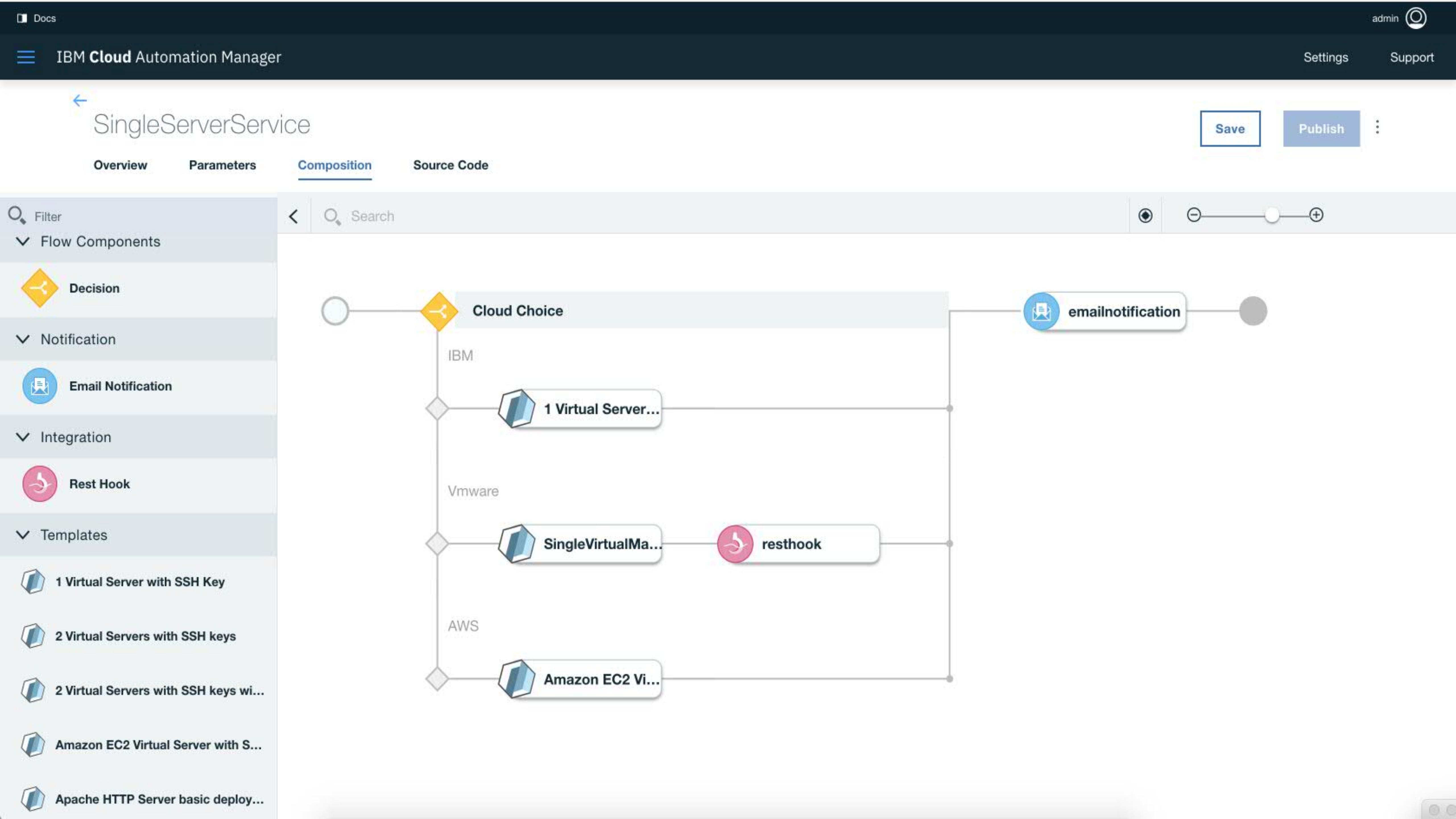
- IBM Public
- RHOCP
- IBM Cloud private

Provider Clouds

- IBM Cloud, VMware
- OpenStack, PowerVC
- AWS EC2, Azure

Services can contain

- Terraform Configurations & variable pre-sets
- Helm Charts
- Simple conditional logic (if/else)
- Notifications (email)
- Order forms



Docs admin

IBM Cloud Automation Manager Settings Support

SingleServerService

Save Publish

Overview Parameters Composition Source Code

Filter

Flow Components

- Decision

Notification

- Email Notification

Integration

- Rest Hook

Templates

- 1 Virtual Server with SSH Key
- 2 Virtual Servers with SSH keys
- 2 Virtual Servers with SSH keys wi...
- Amazon EC2 Virtual Server with S...
- Apache HTTP Server basic deploy...

Search

Terraform variable decoration with camvariables.json

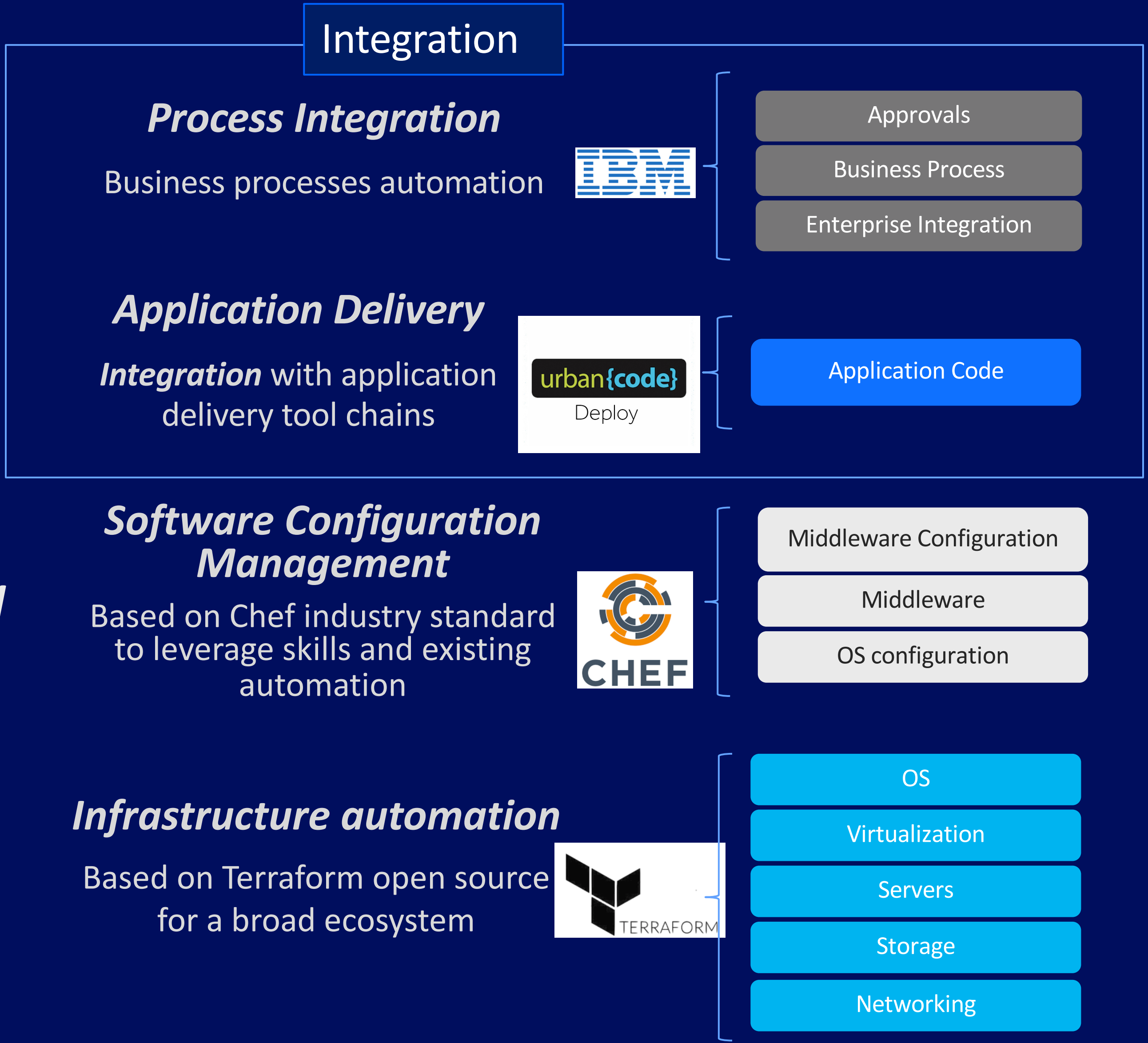
name	Name of the variable in the template
label	Name of variable as displayed in the UI
description	Description of the variable provided in the hover help
type	Type of the variable, affects how is presented in the UI • String, password, counter, boolean, list, map
secured	Does this variable need to be secure on processing and UI needs to mask its content? Default is false
required	Is this variable required? Default is false
hidden	Should the variable be hidden from the deploy dialog? Default is false
immutable	Is the user permitted to modify this variable? Default is false.
default	Indicate the default presented to the user and provided on deployment
regex	Allows user to provide a regular expression to use to validate the input string

Example

```
{
  "name": "cam_pwd",
  "label": "User Password for SSH and MySQL",
  "description": "Password for cam user;
      Allow 8 to 16 alphanum characters",
  "hidden": false,
  "immutable": false,
  "required": true,
  "secured": true,
  "type": "string",
  "regex": "^[0-9A-Za-z]{8,16}$"
}
```

Full stack multi-cloud automation *built with open technology*

- *Automate provisioning of **bare-metal servers, VMs, cloud native services, Docker containers and complex application stacks***
- *Provision workloads with Terraform in **IBM Cloud, VMware, AWS EC2, Azure, PowerVC and OpenStack***
- ***Increase productivity** with a **consistent,** easy to use and easy to understand **user interface***
- *Or use developer friendly scriptable **REST APIs***



Integration

Process Integration

Business processes automation

IBM

Approvals

Business Process

Enterprise Integration

Application Delivery

Integration with application delivery tool chains

urban{code} Deploy

Application Code

Software Configuration Management

Based on Chef industry standard to leverage skills and existing automation

CHEF

Middleware Configuration

Middleware

OS configuration

Infrastructure automation

Based on Terraform open source for a broad ecosystem

TERRAFORM

OS

Virtualization

Servers

Storage

Networking

Catalog of Terraform Content

Starter Library

1. MEAN stack, multi-cloud deployment template
2. LAMP stack, single VM deployment template
3. LAMP stack, multi-cloud deployment template
4. Strongloop Stack on a single VM template
5. Kubernetes cluster with NGINX template
6. MongoDB on a single VM template
7. Strongloop 3 tier deployment template
8. Strongloop Kubernetes cluster template
9. Node.js on a single VM template
10. Amazon EC2 virtual server with SSH key
11. 2 virtual servers with SSH keys, 1 on private network template
12. 1 SoftLayer virtual server with SSH key template
13. 1 Virtual Server with SSH key template
14. 2 Virtual Servers with SSH keys template

Supported OS: CentOS, Ubuntu (used in LAMP and MEAN, in addition to CentOS)
Target cloud: Varies by template

Enterprise Middleware Library

1. Apache HTTP Server Standalone Terraform configuration
2. IBM DB2 Standalone Server Terraform configuration
3. IBM HTTP Server Standalone Terraform configuration
4. WebSphere Liberty Single Server Terraform configuration
5. MQ Basic Single Queue Terraform configuration
6. WebSphere Application Server ND Standalone Server Terraform configuration
7. WebSphere ND Basic Cluster Terraform configuration
8. Oracle MySQL Standalone Terraform configuration
9. Apache Chef Automation Cookbook
10. Apache Tomcat Chef Automation Cookbook
11. Oracle MySQL Community Chef Automation Cookbook
12. MQ Chef Automation Cookbook
13. DB2 Chef Automation Cookbook
14. WebSphere Application Server ND Chef Automation Cookbook
15. WebSphere Liberty Chef Automation Cookbook
16. IBM Installation Manager Chef Automation Cookbook
17. IBM HTTP Server Chef Automation Cookbook

Supported OS: RHEL, Ubuntu (some payloads)
Target: VMWare
Roadmap: IBM Cloud, AWS EC2

Key Components

CAM Components

Key Components

Kubernetes
IBM Cloud Automation Manager is a Dockerized application that runs on IBM Cloud Private (and ICP on RHOCP). Although Cloud Automation Manager is available on DockerHub, Cloud Automation Manager is tightly integrated with IBM Cloud Private and its Kubernetes platform for authorization, role-based access control (RBAC), and other functions. For more information about how to deploy IBM Cloud Private, see IBM Knowledge Center.

Git
Git isn’t bundled as part of Cloud Automation Manager, but a common practice is to store all the templates and content libraries that are created in a Git repository. This practice is also a prerequisite for air-gapped environments that can’t reach github.com where IBM stores its sample templates.

Key Components

Terraform
By default, Cloud Automation Manager manages Terraform only. Terraform is the module in Cloud Automation Manager that communicates with and provisions infrastructure. Cloud Automation Manager comes with many prepackaged Terraform providers, such as IBM Cloud, Amazon Web Services (AWS), Azure, Google Cloud Platform, and OpenStack.

Chef
Chef is a configuration management tool that deploys and configures software from templates, which are also called cookbooks. Chef normally takes over the provisioning after Terraform to start to deploy and configure middleware and applications. Cookbooks for thousands of applications are available from major software vendors on the Chef cookbooks website.
Terraform and Chef overlap in some situations. Cloud Automation Manager segregates responsibilities so that Terraform mostly handles infrastructure and Chef mostly handles configuration and software deployment.

Key Components

Cloud Automation Manager Content Provider (CAMC Provider)
After Terraform provisions the VM, if a Chef template is chosen as part of the deployment, Cloud Automation Manager engages the CAMC Provider. The CAMC Provider is a gateway to a Chef server that takes care of configuration management, middleware, and application deployment.

Helm
Helm charts are to Kubernetes what Terraform templates are to VMs. Helm charts are also a type of infrastructure as code that allows the authoring of Kubernetes resources, such as deployments, services, replica sets, stateful sets, volumes, and ingress rules. With Helm charts, developers can package container information and Kubernetes resources as templates, assign them versions, and deploy them on demand and avoid the tedious tasks of managing the Kubernetes.

Key Components

Cloud Automation Manager Content Provider (CAMC Provider)
After Terraform provisions the VM, if a Chef template is chosen as part of the deployment, Cloud Automation Manager engages the CAMC Provider. The CAMC Provider is a gateway to a Chef server that takes care of configuration management, middleware, and application deployment.

Helm
Helm charts are to Kubernetes what Terraform templates are to VMs. Helm charts are also a type of infrastructure as code that allows the authoring of Kubernetes resources, such as deployments, services, replica sets, stateful sets, volumes, and ingress rules. With Helm charts, developers can package container information and Kubernetes resources as templates, assign them versions, and deploy them on demand and avoid the tedious tasks of managing the Kubernetes.

Key Components

Template library
A Terraform template declares a VM or other cloud resources based on specifications. All the information about the VM is defined there, including the CPU, disk size, and cloud provider. The Template library lists all the templates that are available for provisioning. This library is where administrators create, author, and edit Terraform templates.

Service library
A service is a collection of templates that is run as one process. Cloud Automation Manager can link the provisioning and creation of Cloud Automation Manager templates and IBM Cloud Private Helm charts and provide a way to preset variables and composable execution plans. Within the Service library, the output parameters from one template can be used as an input to another. Administrators create, author, edit, and publish Cloud Automation Manager services to the Service library.

Key Components

Service Composer
The Service Composer is part of the Service library. The Service Composer is the management console in Cloud Automation Manager that allows the authoring of Cloud Automation Manager services. After the Cloud Automation Manager templates are created, administrators can go to the Service Composition section of the Service library and drag the Cloud Automation Manager templates or Kubernetes Helm charts from IBM Cloud Private based on a decided sequence or decision. The Service Composer can also call REST APIs for third-party integration and send emails.

IBM