

Cloud Automation Manager



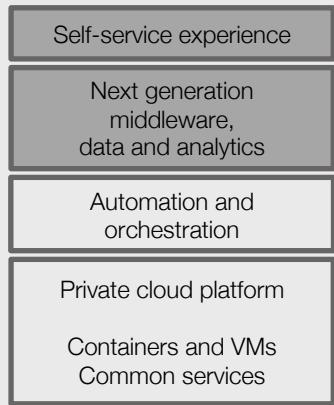
Level Set

The use cases and market forces, the architectural context within the overall IBM Cloud strategy and the functional components of the solution.

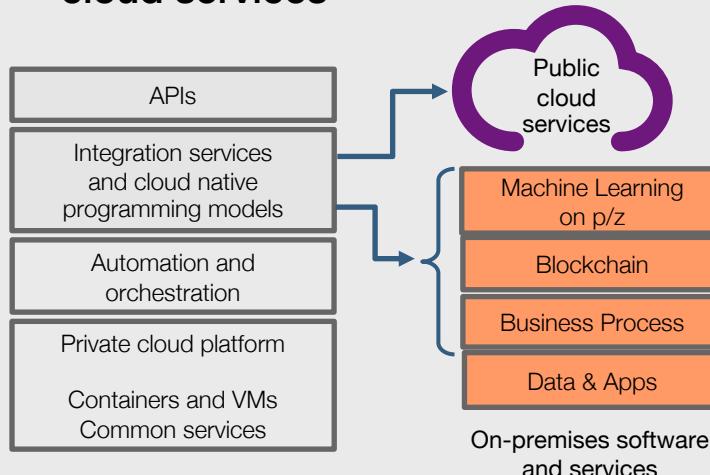


Workload use cases driving private cloud adoption

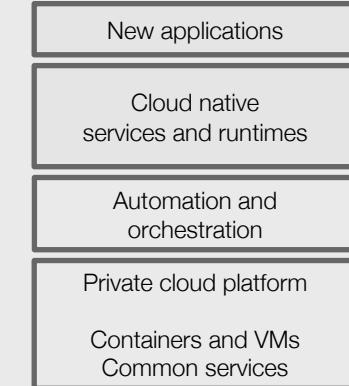
1. Optimize legacy apps with cloud



2. Open your datacenter to work with cloud services



3. Create new cloud native applications



Cloud-enabled middleware

Integration and hybrid cloud

New applications

Match the right workload with the right cloud

Public Cloud

DevOps

ERP

Big data and
analytics

Backup and archive

Front office /
desktop

Risk and
compliance
services

Web applications
and e-commerce

Private Cloud

Mature
workloads

Existing database
workloads

Disaster recovery

Digital
experience
solutions

Customer service

Enterprise
social solutions

Third-party
applications

Mobile
applications

**Maintain
and Evolve**

Applications with
complex processes
and transactions

Applications with
sensitive data

Regulation-intensive
applications

Information-
intensive
applications

Batch processing

Isolated compute
workloads

Development and
test workloads

Non-core business
processes

Not yet virtualized
applications

Highly customized
applications

Business model transformation
required for cloud adoption:

- Strategy
- Design
- Architecture
- Integration
- Processes
- Governance
- Security
- Culture

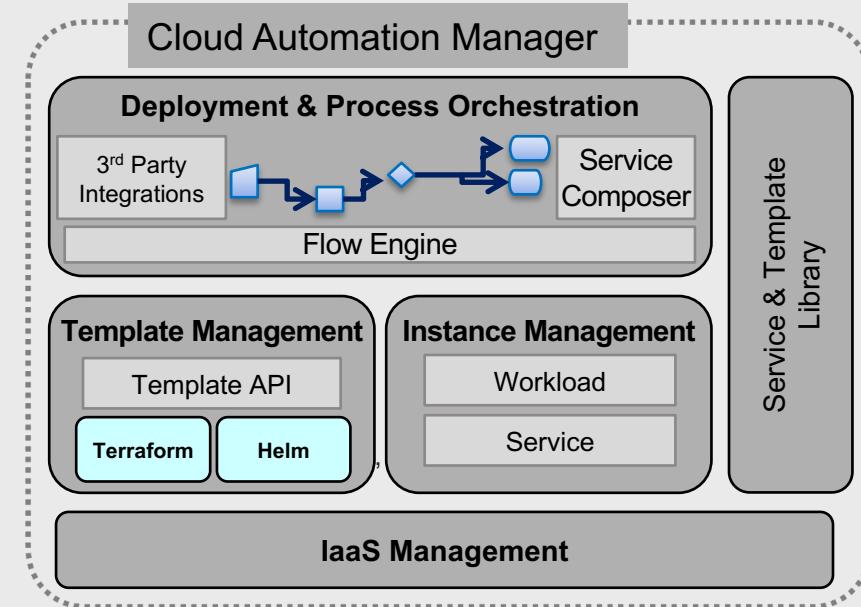
IBM Cloud Automation Manager: Full stack automation and service orchestration

Automated provisioning of infrastructure and applications with workflow orchestration

Self-service access to cloud infrastructure and application services

Manage and govern workloads across multiple and hybrid clouds

Built with **open technology** to avoid vendor lock-in



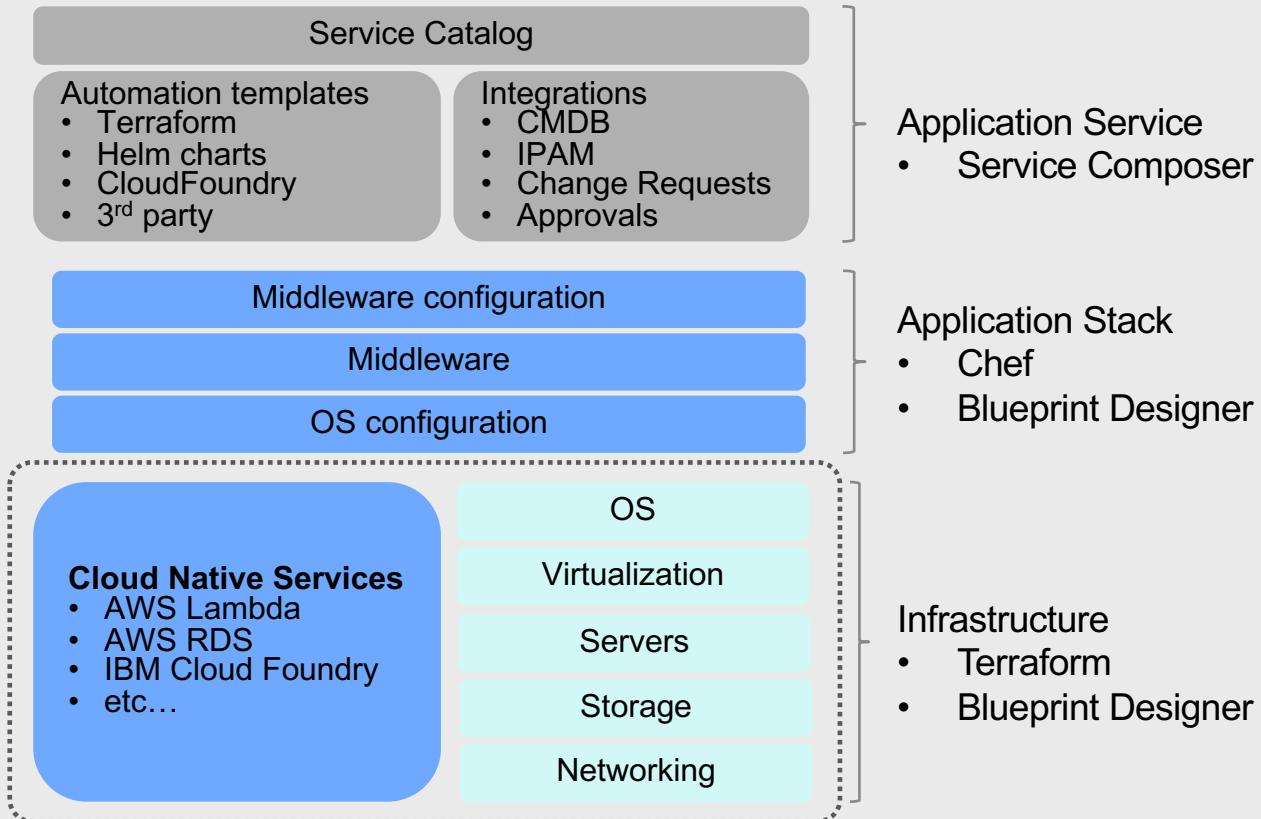
The Cloud Automation Manager business

All clouds, one tool

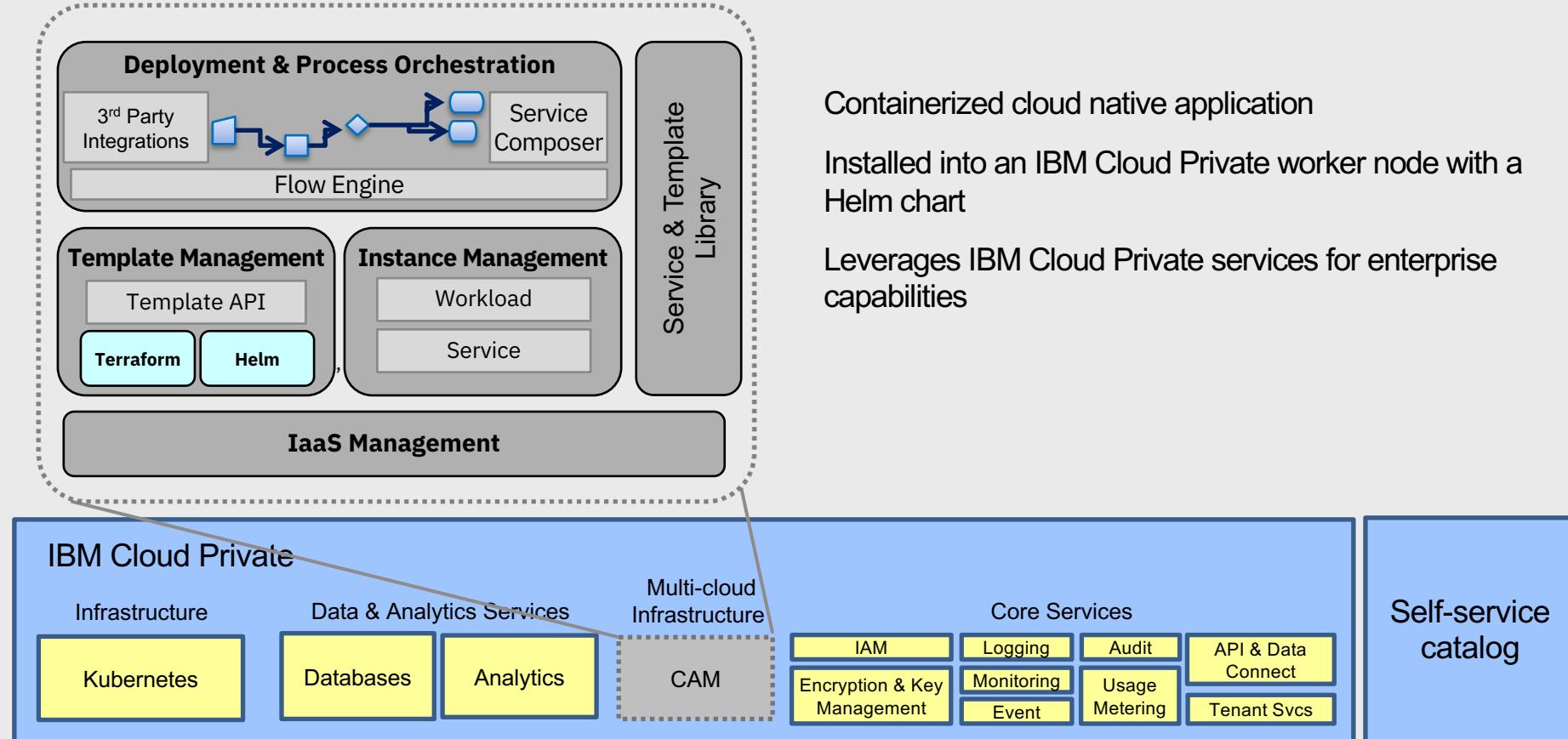
Any workload architecture

Graphical experience

Purpose build for integration



Cloud Automation Manager in IBM Cloud Private



Automation

The ability to automate the provisioning of resources across multiple clouds, including management of deployments in a single pane of glass, with support of a Software Configuration Manager for software install and configuration.

Using Terraform Configurations

1. Select a template

CAM Template Library

Lamp Stack Db2

WAS Liberty MQ

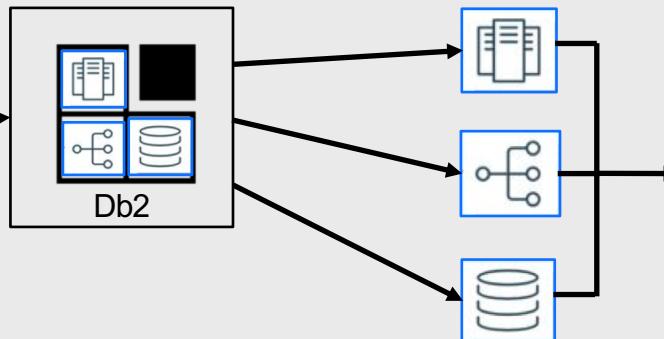
Created by

- You
- Your org
- IBM
- 3rd party

Stored in

- GitLabs
- GitHub Enterprise
- GitHub

2. Review and apply plan



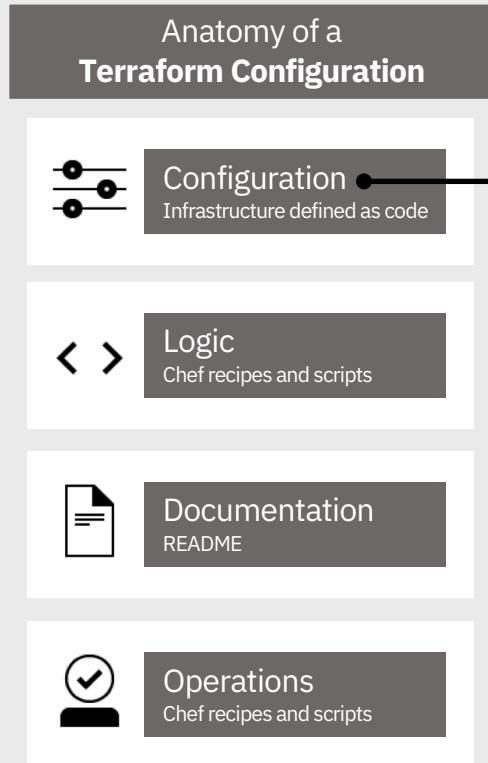
3. Resources are provisioned

4. Use the environment

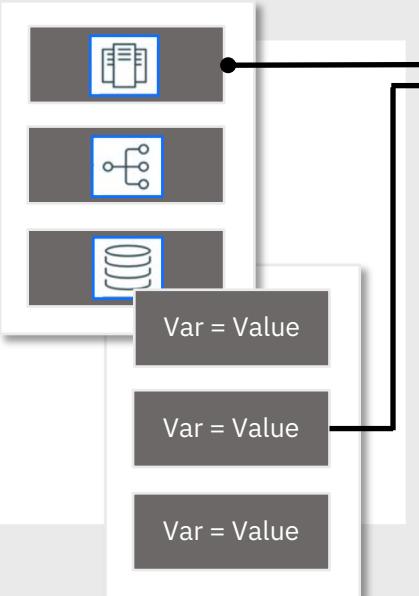
Running instance

```
graph LR; Cloud((Cloud)) --- File[File]; Cloud --- Network[Network]; Cloud --- Database[Database]
```

Anatomy of a Terraform Configuration



Resources
Infrastructure components (compute, network, storage)

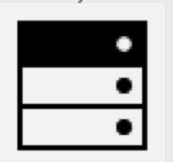


Resource Configuration
For an infrastructure component

```
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 resource parameters

Managing cloud infrastructure as code



```
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}"
  ]
}
```



Store Terraform configuration in Git and manage infrastructure as code

Accelerate development velocity with reusable infrastructure based on open source Terraform

Improve governance and transparency by tracking the ‘who’, ‘what’ and ‘when’ of all environment changes

Improve development team collaboration by enabling team members to easily share application environments

Reduce configuration drift by making it easy to track changes to your running environment

Managing Terraform Providers



```
18 #####
19 #####
20 # Define the vsphere provider
21 #####
22 provider "vsphere" {
23     version      = "~> 0.4"
24     allow_unverified_ssl = true
25 }
26 #####
27 #####
```

How to use a specific provider?

1. **Include version attribute on provider section** this will let indicate which version of the provider is use on the template
2. **Provider downloaded from internet** if the version requested is not included in the system.
3. **Allows user to bring newer versions or add new provider** by downloading the provider and loading it into a specific volume used by CAM microservices.

Shared Data Objects

Administration Objects that allow configuration data from External system that services or templates will interact or need the configuration of that environment

- Example – remote database or LDAP server, Cloud configuration (Vcenter datacenter, storage, network, others)

Template defines the dependency on a data type

- System finds the matching list of objects
- User select one

Template metadata camvariables.json

Define dependency on datatype

```
{  
    "input_datatypes": [  
        {  
            "name": "kubeconfig",  
            "label": "Kube config"  
        }  
    ],  
    "output_variables": [  
        {  
            "name": "cluster_config",  
            "label": "Cluster configuration",  
            "type": "string",  
            "description": "Base64 encoded content of the kubeconfig yaml file.",  
            "secured": true,  
            "required": true,  
            "hidden": true,  
            "default": "${kubeconfig.cluster_config}",  
            "immutable": true  
        }  
    ]  
}
```

Define mapping of datatype attribute to variables

```
{  
    "input_datatypes": [  
        {  
            "name": "kubeconfig",  
            "label": "Kube config",  
            "type": "string",  
            "description": "Base64 encoded content of the kubeconfig yaml file.",  
            "secured": true,  
            "required": true,  
            "hidden": true,  
            "default": "${kubeconfig.cluster_config}",  
            "immutable": true  
        }  
    ],  
    "output_variables": [  
        {  
            "name": "cluster_config",  
            "label": "Cluster configuration",  
            "type": "string",  
            "description": "Base64 encoded content of the kubeconfig yaml file.",  
            "secured": true,  
            "required": true,  
            "hidden": true,  
            "default": "${kubeconfig.cluster_config}",  
            "immutable": true  
        }  
    ]  
}
```

Template lifecycle – Plan/Apply

Allows user to make changes on running instances

Two major scenarios:

1. User want to change a parameter on an instance
 - Example – change memory/cpu
2. User wants to update to a new version of the template used to create their instance
 - Example – add more VMs or other components

Instance modification/update in a two-step process:

1. Plan - allows user to select the version and /or change parameters and execute a phase to see what will change in the environment
2. Apply – executes the changes presented on plan action on the environment

Plan/Apply

1. Select template version

Deployed Instances
test33333-00-instance • Running

Overview Modify Log File

Modify Deployed Instance

Selecting a new version might add additional parameters to your deployment, it might also remove some existing values. Please check the template version source for further info.

Select Version

Template Version demo

demo (current)

Next

2. Modify parameters and execute Plan

Deployed Instances
test33333-00-instance • Running

Overview Modify Log File

Modify Deployed Instance

Select Version Edit Parameters Apply Changes

Back Plan Changes

input_group

Cloud Connection

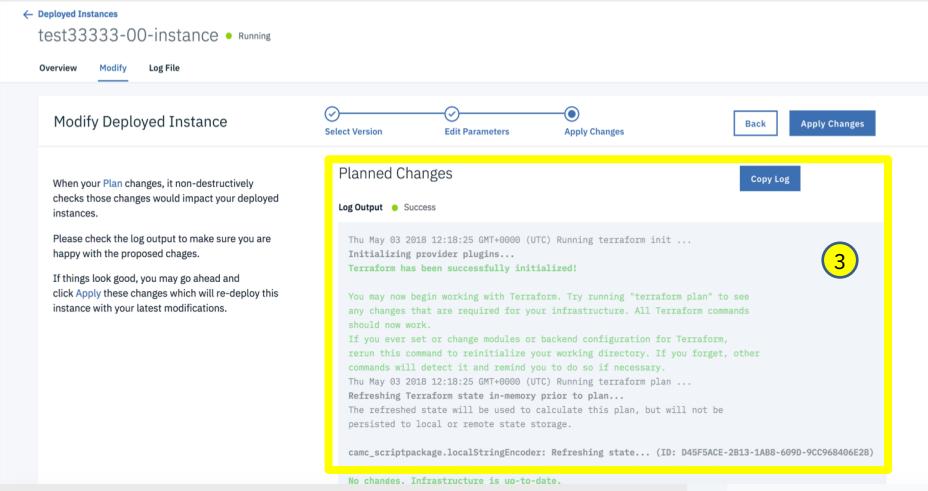
Other

input_group

• Input String: String 2 value

• Input String: Encoder

Plan/Apply (continued)



← Deployed Instances
test33333-00-instance • Running

Overview Modify Log File

Modify Deployed Instance

Select Version Edit Parameters Apply Changes Back Apply Changes

When your Plan changes, it non-destructively checks those changes would impact your deployed instances.

Please check the log output to make sure you are happy with the proposed changes.

If things look good, you may go ahead and click Apply these changes which will re-deploy this instance with your latest modifications.

Planned Changes

Log Output • Success

Thu May 03 2018 12:18:25 GMT+0000 (UTC) Running terraform init ...
Initializing provider plugins...
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

Thu May 03 2018 12:18:25 GMT+0000 (UTC) Running terraform plan ...
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be persisted to local or remote state storage.

camc_scriptpackage.localStringEncoder: Refreshing state... (ID: D45F5ACE-2B13-1A88-6090-9CC968406E28)

No changes. Infrastructure is up-to-date.

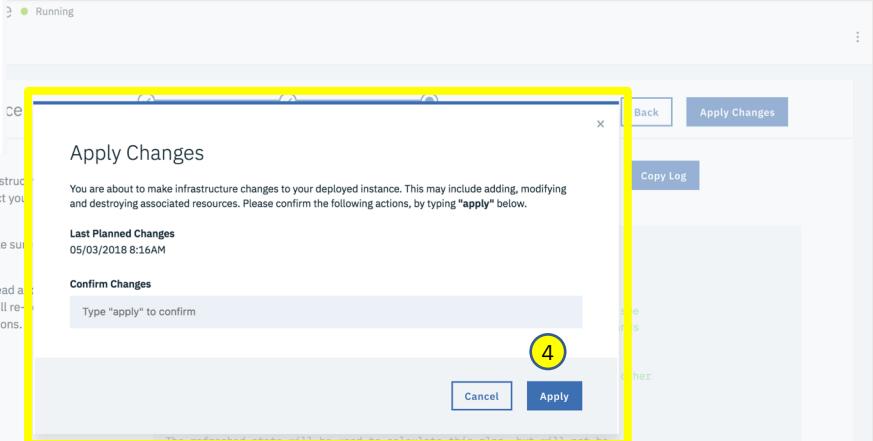
When your Plan changes, it non-destructively checks those changes would impact your instances.

Please check the log output to make sure you are happy with the proposed changes.

If things look good, you may go ahead and click Apply these changes which will re-deploy this instance with your latest modifications.

4. Apply the changes

3. Verify the plan output



• Running

Apply Changes

You are about to make infrastructure changes to your deployed instance. This may include adding, modifying and destroying associated resources. Please confirm the following actions, by typing "apply" below.

Last Planned Changes
05/03/2018 8:16AM

Confirm Changes

Type "apply" to confirm

Cancel Apply

The following changes will be made to your infrastructure. Terraform will not persist changes to your real infrastructure until you apply them.
camc_scriptpackage.localStringEncoder: Refreshing state... (ID: D45F5ACE-2B13-1A88-6090-9CC968406E28)

No changes. Infrastructure is up-to-date.

Creating Terraform templates – CAM Template Designer

Graphical and text-based Terraform template designer

Built-in integration with Git to easily share and re-use templates

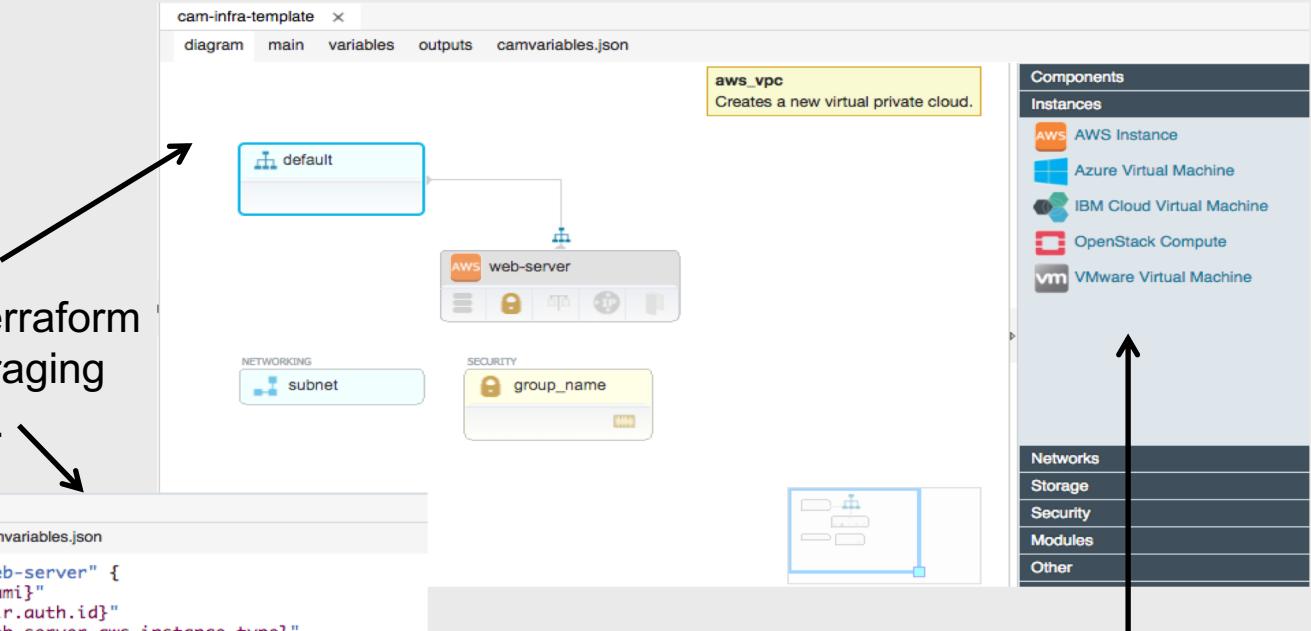
Rich cloud support to create complex Terraform content via drag and drop

Easily create and publish CAM templates

Integration with UrbanCode Deploy to automate full-stack application deployments



CAM Template Designer – Terraform Design



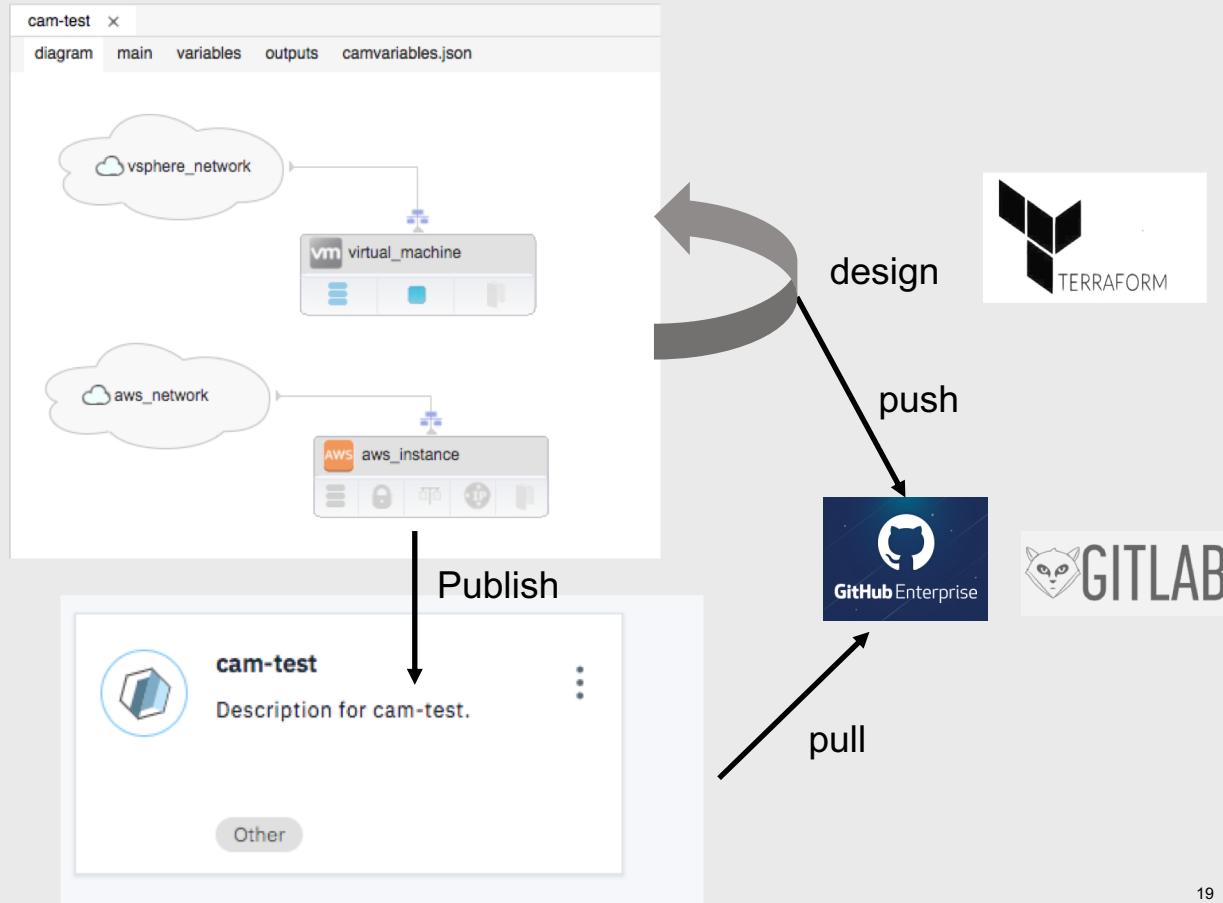
Graphically create Terraform templates while leveraging Terraform constructs.

```
cam-infra-template ×  
diagram main variables outputs camvariables.json  
19resource "aws_instance" "web-server" {  
20  ami = "${var.web-server_ami}"  
21  key_name = "${aws_key_pair.auth.id}"  
22  instance_type = "${var.web-server_aws_instance_type}"  
23  availability_zone = "${var.availability_zone}"  
24  subnet_id = "${aws_subnet.subnet.id}"  
25  vpc_security_group_ids = ["${aws_security_group.group_name.id}"]  
26  tags {  
27    Name = "${var.web-server_name}"  
28  }  
29}  
30  
31resource "aws_vpc" "default" {  
32  cidr_block      = "0.0.0.0/0"  
33  enable_dns_hostnames = true  
34  tags {  
35    Name = "${var.network_name_prefix}"  
36  }  
37}
```

Rich cloud support

CAM Template Designer – Create CAM Templates

1. Create CAM project
2. Edit Terraform
3. Publish to CAM



CAM Template Designer – UrbanCode Deploy Integration

UCD Terraform provisioner

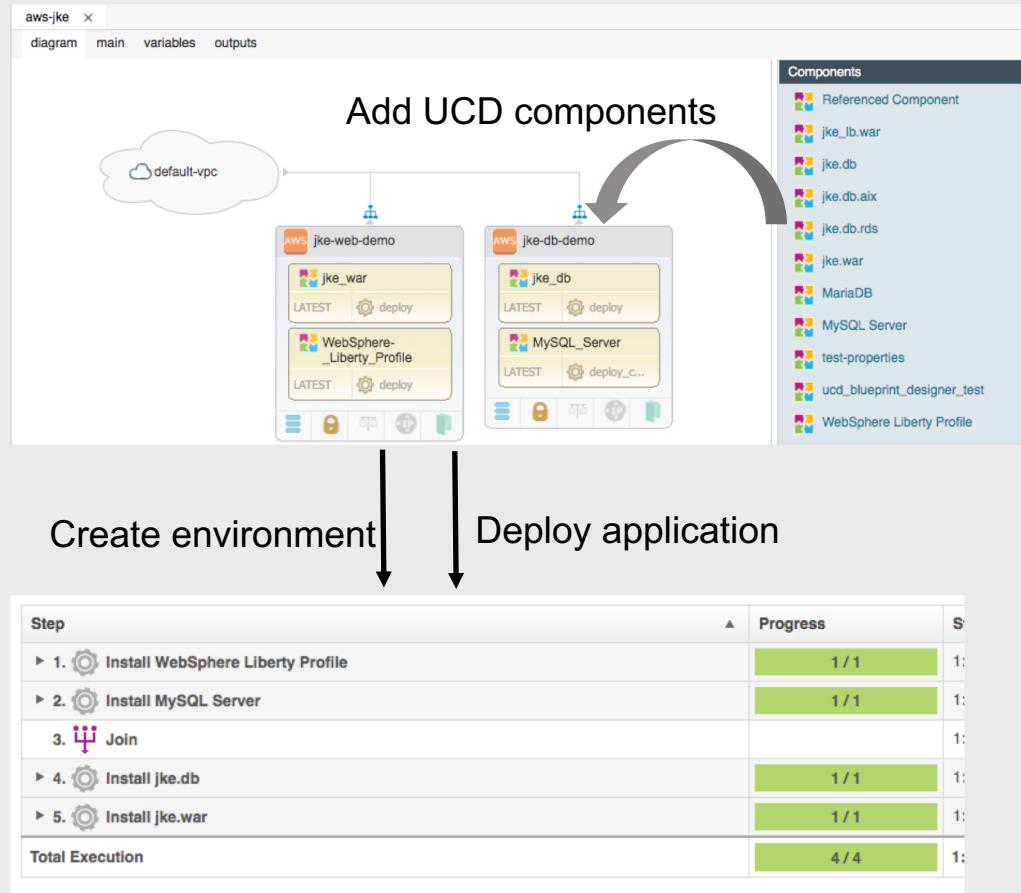
- Install UCD agent
- Included in CAM Terraform runtime

UCD Terraform provider

- Resources to automate UCD
- Create resource tree, environment, mappings
- Execute application and component processes
- Included in CAM Terraform runtime

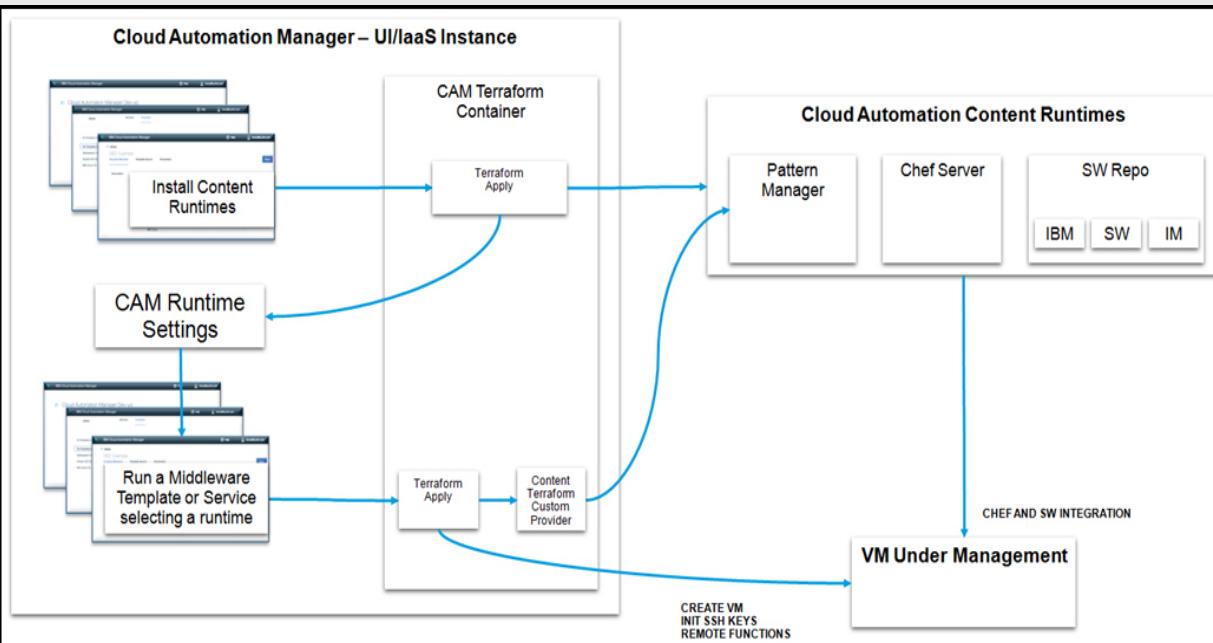
UCD Designer integration

- View UCD components in palette
- Drag and drop components onto diagram
- Automatically create UCD Terraform resources



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

You supply the virtual server, let CAM stand-up a pre-configure Chef runtime

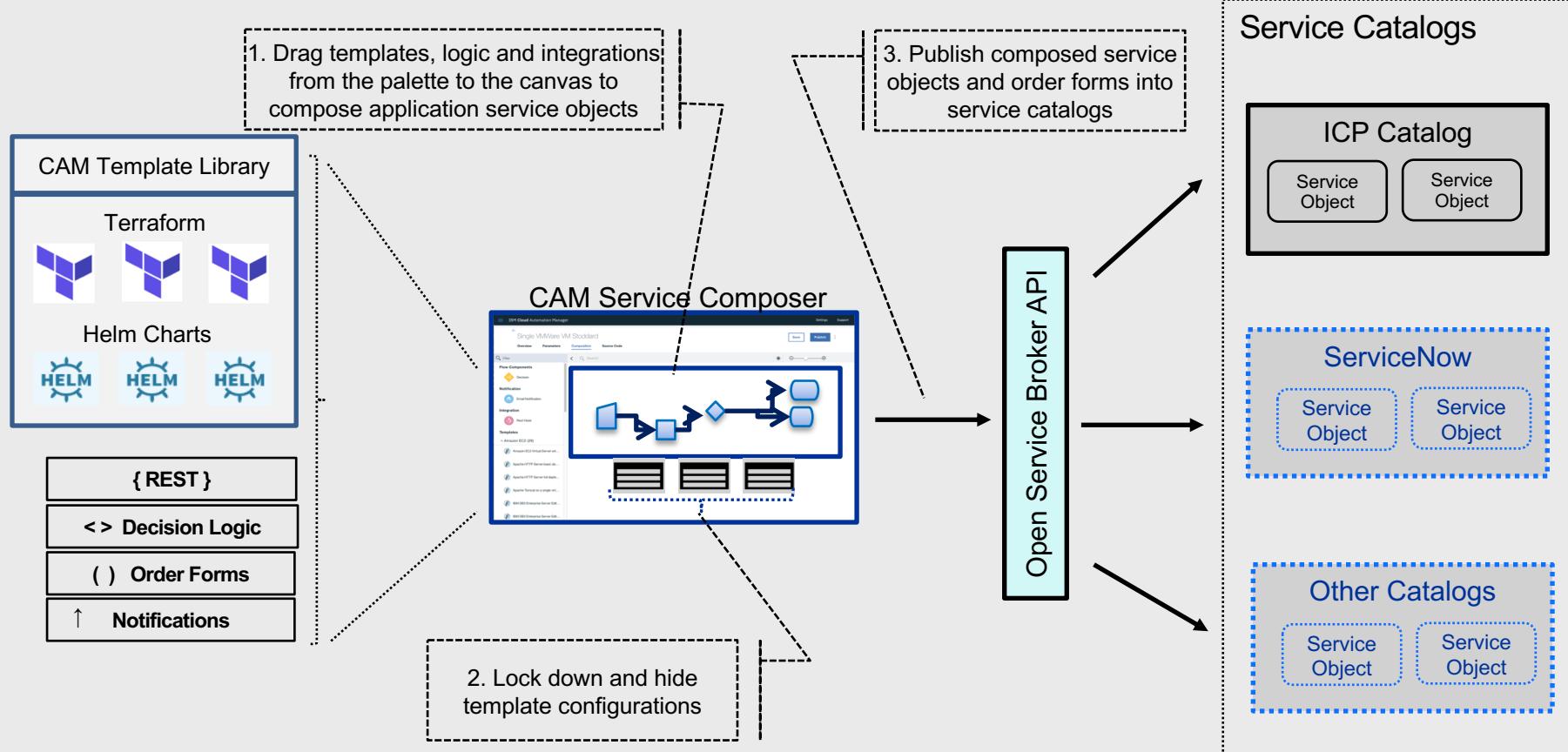
Fully containerized for maximum portability

Integrated software repository

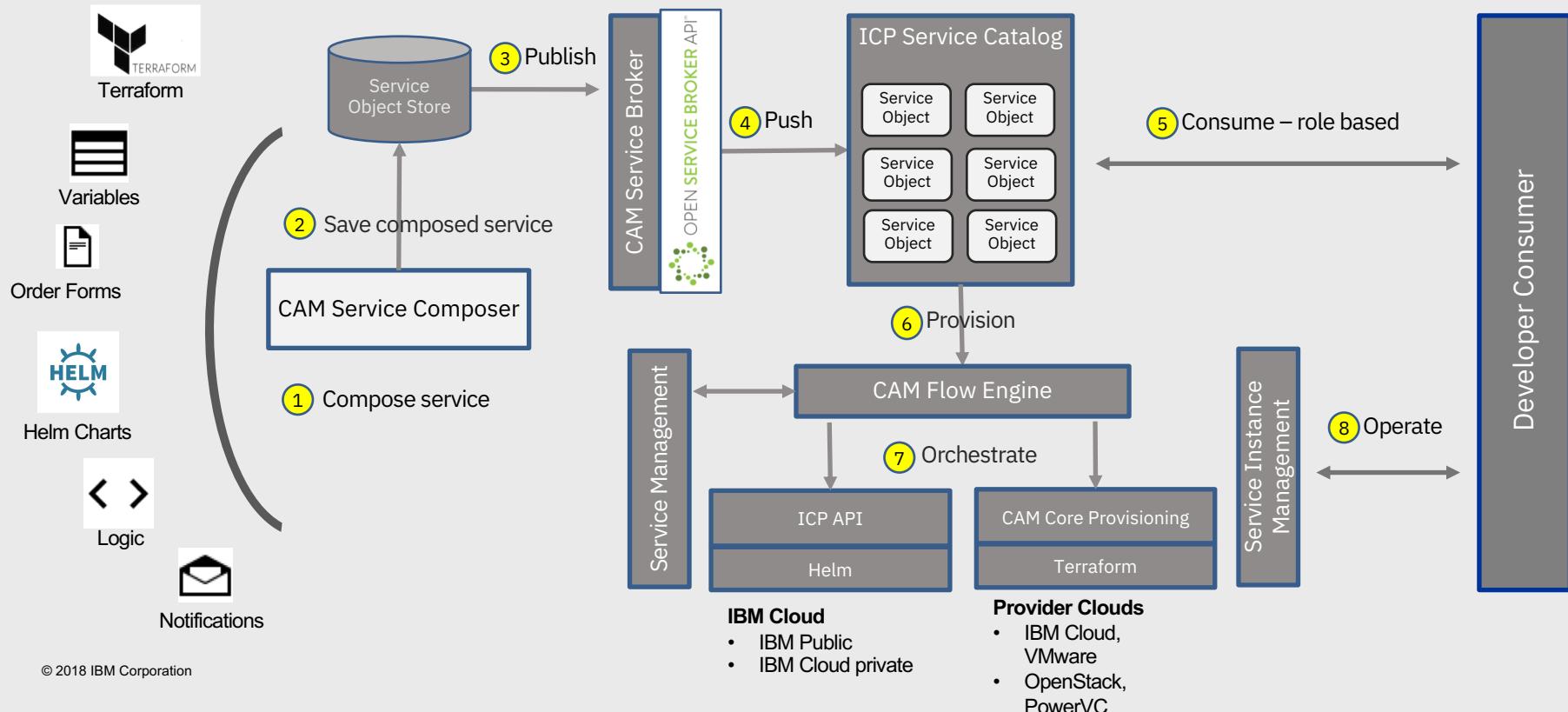
Orchestration

Assembling automations into a flow that represents the end-to-end lifecycle of a service and supports publication into self service catalogs.

Service Composition and Flow Engine



Service Management Lifecycle: Compose, Publish, Consume and Operate



Assemble, Curate, Publish - with Service Composer

IBM Cloud Automation Manager Docs Support

WASNDService

Overview **Composition** Parameters Plans & Form Source Code

Filter

Flow Components

- Decision
- Notification
- Email Notification
- Integration
- Rest Hook

Templates

Helm

1. **Assemble** the service by dragging activities from the palette and connecting on the canvas. This service defines three plans.

2. **Curate** the service by locking down configuration variables

3. **Publish** the service into the ICP Catalog

Save Publish

IBM WebSphere Network Deployment V9 on a single ... X

Basic Information Parameters

Search Parameters

INPUT PARAMETERS	VALUE
WASN01_dns_servers	1 Items
WASN01_dns_suffixes	1 Items
WASN01_domain	<code>\$(templates.infobloxc35d464.output.associated_domain)</code>
WASN01-image	Content/ContentRH_Template_2018_1Q
WASN01-os_admin_user	root
WASN01-os_password	Op3nPatters
user_public_ssh_key	None
WASN01_root_disk_size	100
WASN01-name	<code>\$(templates.infobloxc35d464.output.associated_</code>

Example: WAS ND Service defining overview for catalog

IBM Cloud Automation Manager

Docs Support

WASNDService

Save Publish :

All changes saved.

Overview Composition Parameters Plans & Form Source Code

Quick Overview

* Service Display Name
WASNDService

Short Description

This service is used to deploy a VM on a cloud with Websphere Application Server software installed. It will also perform an update in CMDB.

139/150

Detail Overview

Long Description

This service is used to deploy a VM on a cloud with middleware software installed. As per user selection application will get deployed and CMDB will get updated with server records.

Features

- IPAM Server
- IBM WebSphere Network Deployment
- Helm Chart for ELK deployment
- ELK agent Installation
- APM Agent Installation
- CMDB update

Example: Mapping parameters

IBM Cloud Automation Manager

WASNDService

Save Publish : All changes saved.

Overview Composition Parameters Plans & Form Source Code

Service Parameters

PARAMETER KEY	DEFAULT VALUE	END-USER PERMISSION	DISPLAY NAME	PARAMETER TYPE	ACTIONS
Environment	[{"value": "Development", "label": "Development"}, {"value": ...	Read-Write	Environment	string	⋮
Receiverlist		Read-Write	Receiver list	string	⋮
WASNode01_root_disk_size		Read-Write	WASNode01_root_disk_size	string	⋮

Create Parameter

Activity Parameters

TEMPLATE NAME	PARAMETER KEY	DEFAULT VALUE	ACTIONS
IPAM			⋮
IBM WebSphere Network Deployment V9 on a single virtual machine			⋮
ibm-charts/bm-iclogging-kibana			⋮
ELK Agent Installation Template			⋮
ibm-charts/bm-iclogging-kibana			⋮
ELK Agent Installation Template			⋮
APM Agent Installation Template			⋮
APM Configuration for WebSphere Application Server Template			⋮
Service Now CMDB Configuration Template			⋮

Edit Parameter

Parameter Key: WASNode01_dns_servers
Parameter Type: list

Update Link Create New

You can link to refer the value of a service parameter or template parameter
Note : The linked parameter should be of the same type as that of the parameter you are editing

Current Value: ["9.5.46.5"]

Search Parameters

- > Service Parameters 3 parameters
- > IPAM 8 parameters

Cancel Save

Example: Defining Plans

IBM Cloud Automation Manager

Docs Support

WASNDService

Save Publish :

All changes saved.

Overview Composition Parameters **Plans & Form** Source Code

Plans i

Add Plan

NAME	DESCRIPTION	ACTIONS
Standard	To deploy a Standard plan	⋮
Development	In case of development "WASNode01_root_disk_size" is 250 GB.	⋮
Test	In case of Test "WASNode01_root_disk_size" is 350 GB.	⋮
Production	In case of Production "WASNode01_root_disk_size" is 500 GB.	⋮

Plan Parameters i

PARAMETER KEY	DEFAULT VALUE	END-USER PERMISSION	DISPLAY NAME	TYPE	ACTIONS
Standard					
Development					
Test					
Production					

Example: WAS ND Service in the ICP Service Catalog



ibm-transadv-dev

IBM Cloud Product Insights Transformation Advisor



ibm-transadv-dev

IBM Cloud Product Insights Transformation Advisor



ibm-voice-gateway-dev

IBM Voice Gateway Helm chart (Developer Trial)

ibm-charts

ibm-charts-internal

ibm-charts



ibm-websphere-liberty

WebSphere Liberty for Linux on amd64, ppc64le and s390x



ibm-websphere-liberty

WebSphere Liberty for Linux on amd64, ppc64le and s390x



ibm-webterminal-dev

A browser-based full xterm terminal.

ibm-charts

ibm-charts-internal

ibm-charts



ibmcloudcloudantservice

ibmcloudcloudantservice



icsservice

icsservice



wasapplicationservice

wasapplicationservice

service

service

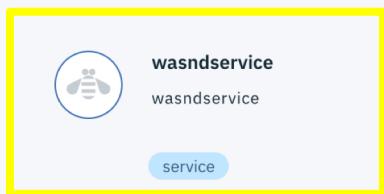
service



wasndservice

wasndservice

service



Example: Ordering the WAS ND Service from the ICP Service Catalog

IBM Cloud Private Create resource Docs Support 

[View all](#)

wasndservice

wasndservice WASNDService

VERSION 2458111
PUBLISHED Mar 7th 2018
TYPE Service

Details

Helps user deploy WAS ND in multiple formats and connected to different systems

Useful Links

Documentation:
Support link:

The user can select from three service plans

Plans

PLAN	FEATURES	PRICING
development	To deploy a Standard plan	Free
staging	Staging	Free
production	Production	Free

[Configure](#)

Integration

Integrating Cloud Automation Manager as part of
IBM Cloud Private with external components within
the datacenter.

Integration Strategies with Composer

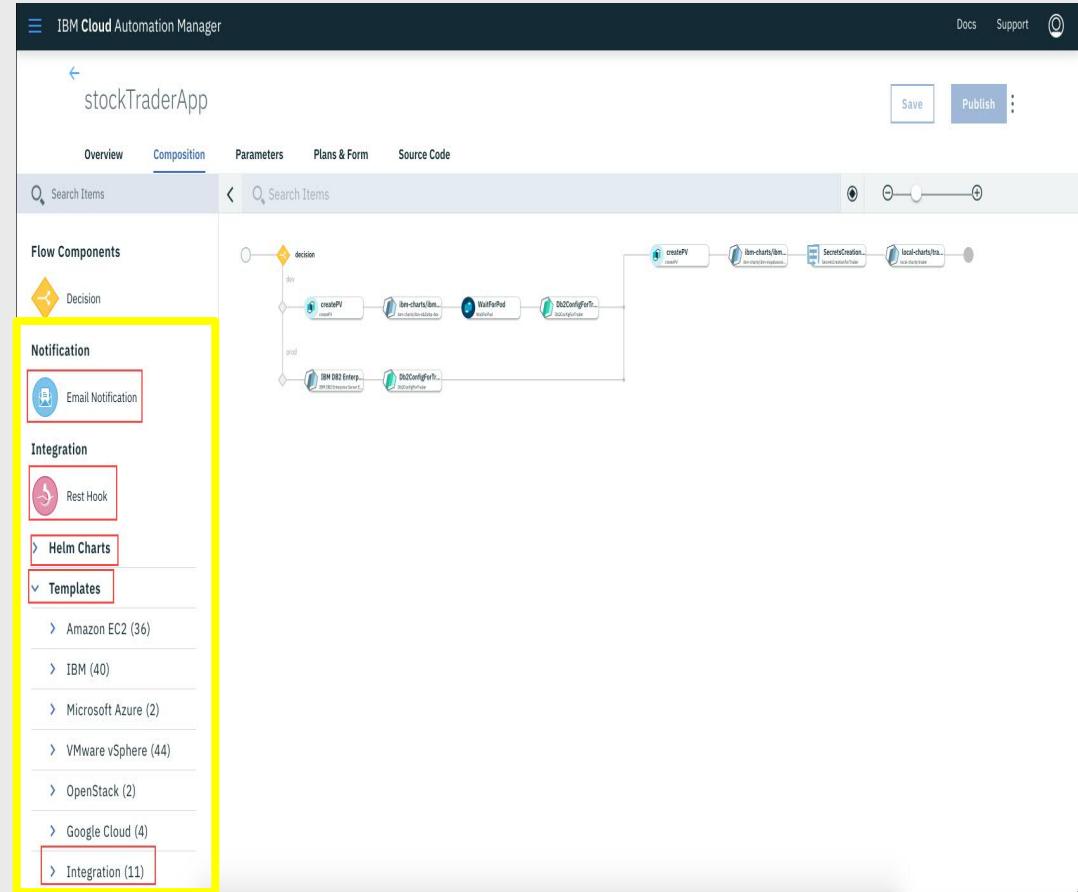
Email notifications to SMTP gateways

REST calls to any external systems

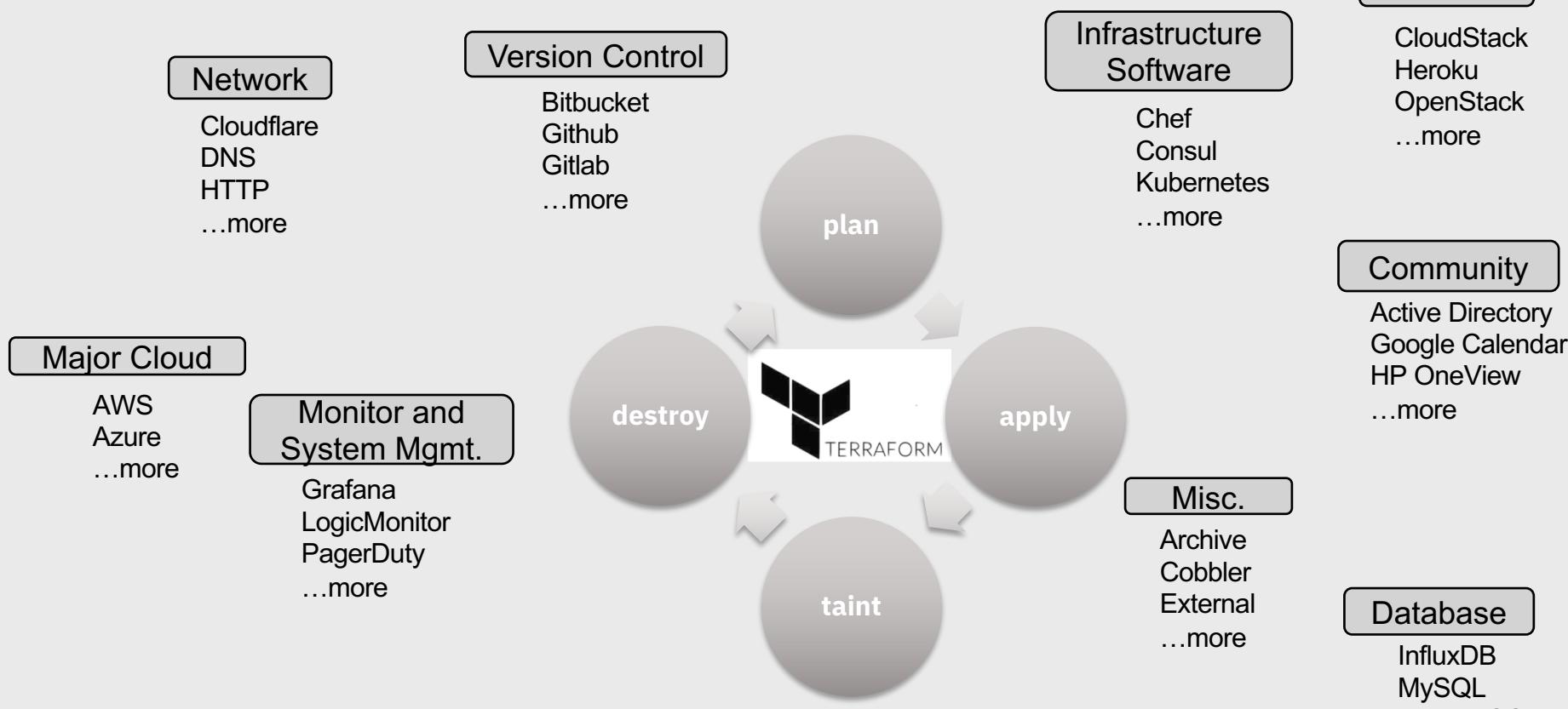
HELM deployment on IBM Cloud Private

Terraform templates on different cloud as IAAS

Terraform templates as scripts on remote / local systems



CAM Integration via Terraform provider eco-system



Resources

CAM Knowledge Center

<https://www.ibm.com/support/knowledgecenter/SS2L37>

CAM Developer Portal (Blogs, Tutorials, Videos)

<http://developer.ibm.com/cloudautomation>