Architecting The Agile Enterprise With The Agile Cloud Manager

By Jim Slavin, M.S.
Principal Consultant
Agile Cloud Institute, Inc.

Introduction

And Table of Contents

The Agile Cloud Manager can make it very easy for you to extend your agile project management to include all your infrastructure-as-code, all your configuration-as-code, and all your SaaS definition templates, so that your platforms can iteratively evolve to enable rapid implementation of your corporate-level initiatives.

This white paper will explain the high-level architecture for IT leadership and for Architectural Review Boards.

The sections of this white paper are as follows:

Use Cases

The use cases for the Agile Cloud Manager are summarized in the following list:

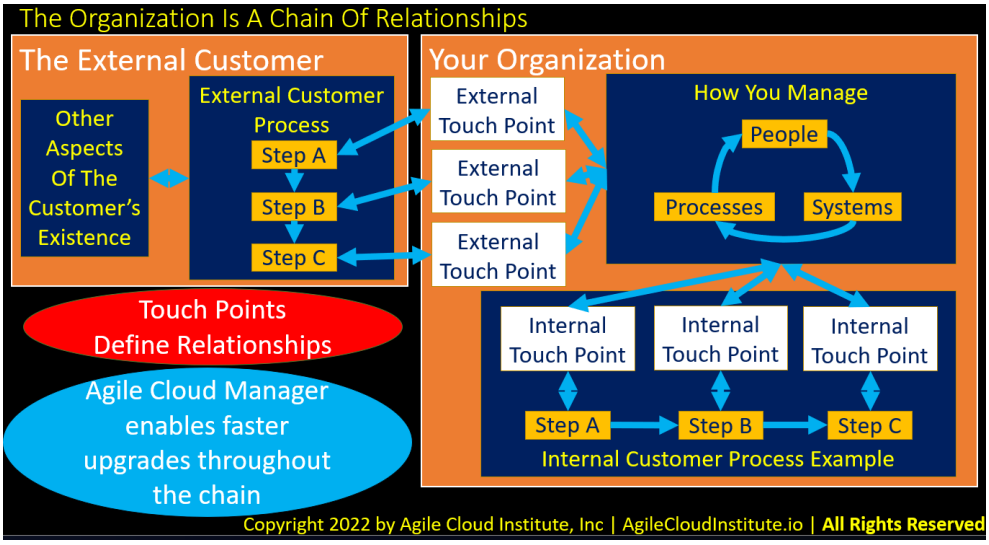
- Upgrade multiple applications in parallel to accelerate customer value creation, increasing the speed with which new features can be implemented
- Failover and Failback between different cloud providers
- Reduce redundant engineering efforts
- Encapsulate security components
- Create cloned environments quickly
- Manage business-level components more easily
- Integrate all engineering work into a single enterprise pipeline
- Put every aspect of enterprise value stream under agile project management
- Simplify governance, security, and compliance
- Integrate any pipeline tool with infrastructure-as-code and with configuration-as-code
- Reduce the amount that engineers are required to learn

At the end of this article, a table will explain how each of these use cases is fulfilled by specific aspects of the Agile Cloud Manager.

But for now, let's explain what the Agile Cloud Manager is and how it relates to your organization so that you will be better able to understand the use cases table at the end of this article.

The Organization is a Chain Of Relationships

The first step towards deriving all the benefits of the Agile Cloud Manager involves looking at your entire organization as a chain of relationships.



As you can see from the following diagram entitled "The Organization Is A Chain Of Relationships", the work of your organization involves connecting external customers with internal customers.

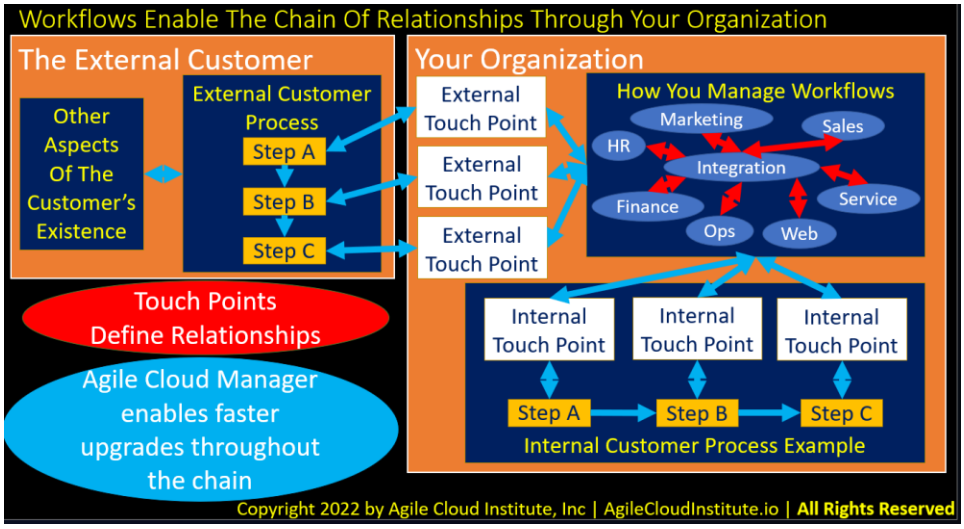
These external and internal customers interact with your organization while performing specific processes that have understandable steps. And each step in each process requires that your organization provide specific touch points through which the work of each step is coordinated.

Each of your organization's touch points is defined in terms of how you manage people, processes, and systems.

Workflows are the way that your organization can define and evolve the ways that you manage people, processes, and systems.

The Agile Cloud Manager makes it a lot easier to deploy upgrades throughout the chain of workflow relationships so that your organization can increase customer-derived value in a more agile way.

The next flowchart, entitled "Workflows Enable The Chain Of Relationships Through Your Organization" adds a "How You Manage Workflows" section to the first flowchart to better illustrate how your organization's workflows pass through each of the integrated IT services which together define your enterprise-level platform.

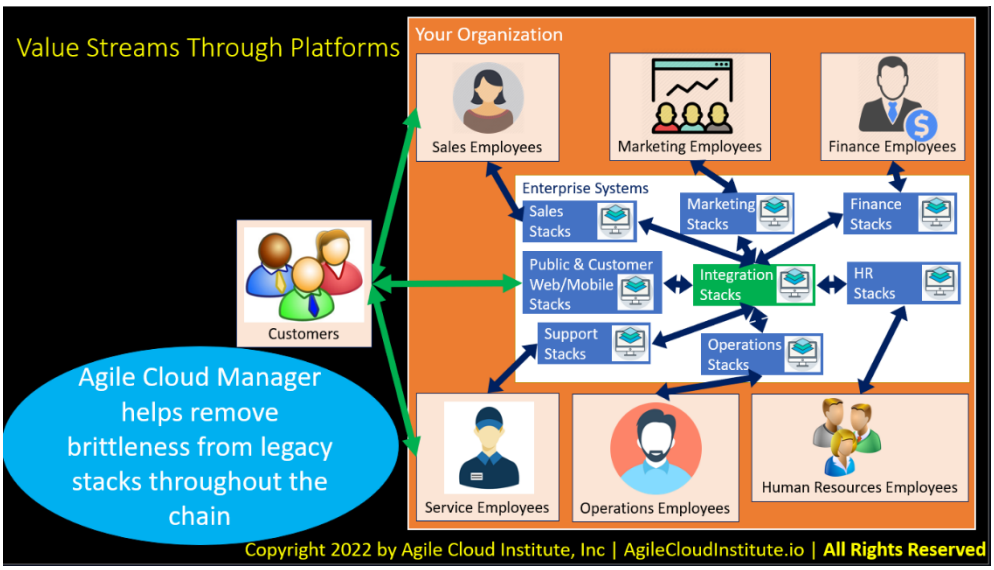


As you can see from the new "How You Manage Workflows" section in the second diagram, your enterprise-level platform is decomposed into sub-platform buckets for major functions such as sales, marketing, service, web, finance, human resources, and the backend fulfillment of value, which here we are referring to as Ops or operations.

Value Streams Through Platforms

To make sense of your workflows, it helps to look deeper into the sub-platform buckets from the preceding diagram and to consider how different types of people interact with each type of sub-platform.

The third diagram is entitled "Value Streams Through Platforms" and illustrates various types of people interacting with the stacks that your organization uses to manage each type of sub-platform.



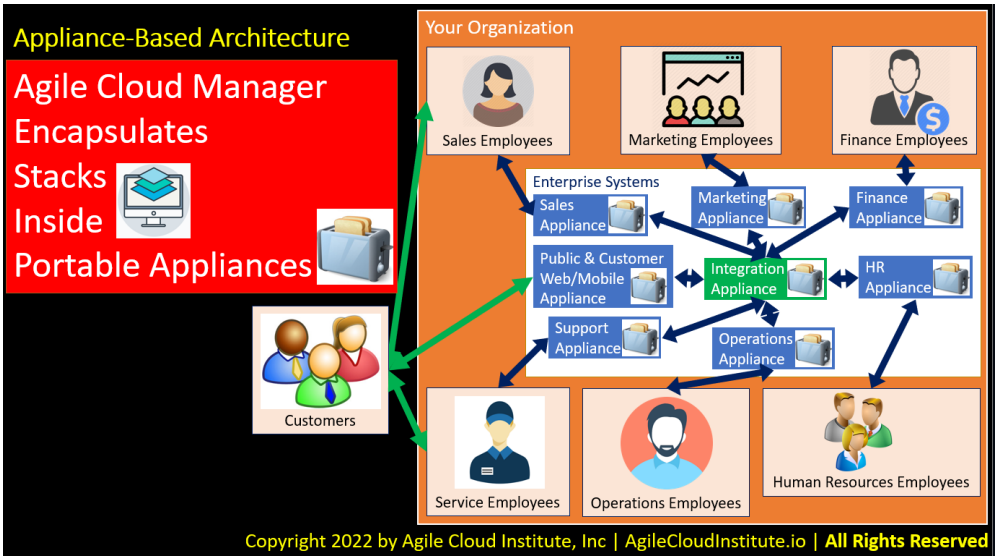
In the old days, stacks were brittle, cumbersome things that required a lot of high maintenance care and feeding. Even today, many legacy organizations have legacy stacks that are not yet under agile management, and which are not yet software-defined.

Appliance-Based Architecture

The Agile Cloud Manager enables an organization to encapsulate its services inside simple appliances that are cloud-agnostic and that can be managed using one simple interface.

The fourth diagram is entitled "Appliance-Based Architecture" and illustrates how simple appliances make it so much easier for your systems to be managed.

An appliance should be as simple to use as a toaster.



This way, your applications teams can limit their involvement to configuring toasters, and your platform/infrastructure teams can focus on repeatable processes for delivering high quality toasters.

The Agile Cloud Manager Simplifies Appliances

To make it easier for your teams to upgrade your platform to become a collection of easily-managed appliances, the Agile Cloud Manager uses simple Command Line Interface CLI commands and simple configuration files that make it easy to plug the Agile Cloud Manager into any pipeline tool.

This means that your systems can be controlled interchangeably by Jenkins, Azure Pipelines, CircleCI, GitHub, or any other pipeline tool. The ability to switch back and forth between any pipeline tool is an essential feature which enables you to avoid vendor lock-in while also maximizing the flexibility with which you can manipulate infrastructure and platforms.

The fifth diagram is entitled "Agile Cloud Manager Simplifies Appliances".

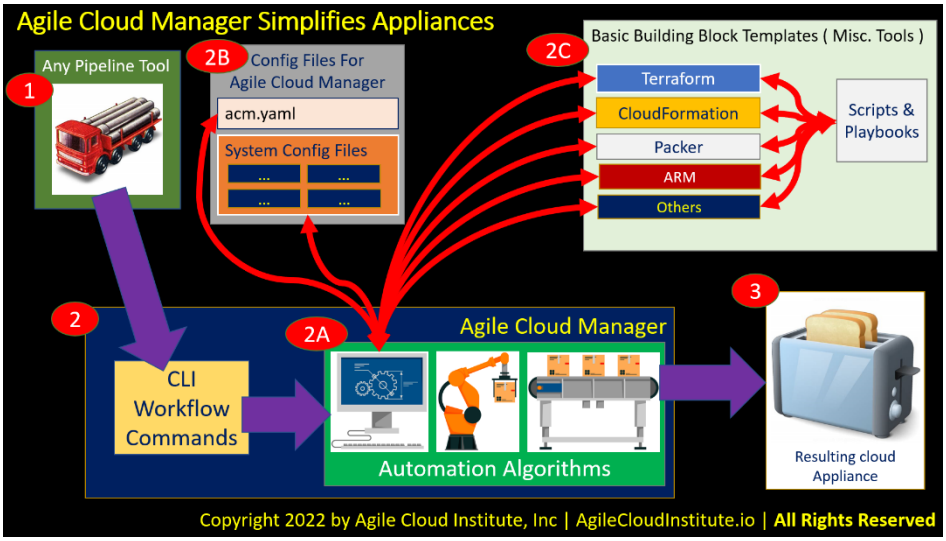


Diagram 5 shows how the Agile Cloud Manager includes automation algorithms that you can control using pipeline tools that call simple CLI workflow commands.

Diagram 5 also shows that you can use simple configuration files to control how those simple CLI workflow commands are executed. What is more, Diagram 5 also illustrates how your infrastructure templates then get operated on by the algorithms.

This architecture enables you to separate out different types of code into different building blocks to maximize reusability and to improve the quality and reliability of each reusable element.

You can swap out elements and you can even completely change cloud providers without ever touching your application code because this architecture enables your infrastructure and platform elements to remain safely encapsulated behind a simple interface.

The result is a simple, consistent appliance with which your people and systems can interact easily without ever needing to know where the system is hosted, or which cloud provider is being used as a vendor for the hosting.

The Enterprise Pipeline Builds Up From Simple, Cloud-Agnostic Interfaces

The sixth diagram, entitled "Enterprise Pipeline Builds Up From Simplified Interfaces", shows how the Agile Cloud Manager can create appliances by reusing your pre-existing building blocks to compose increasingly higher-level components of your enterprise systems.

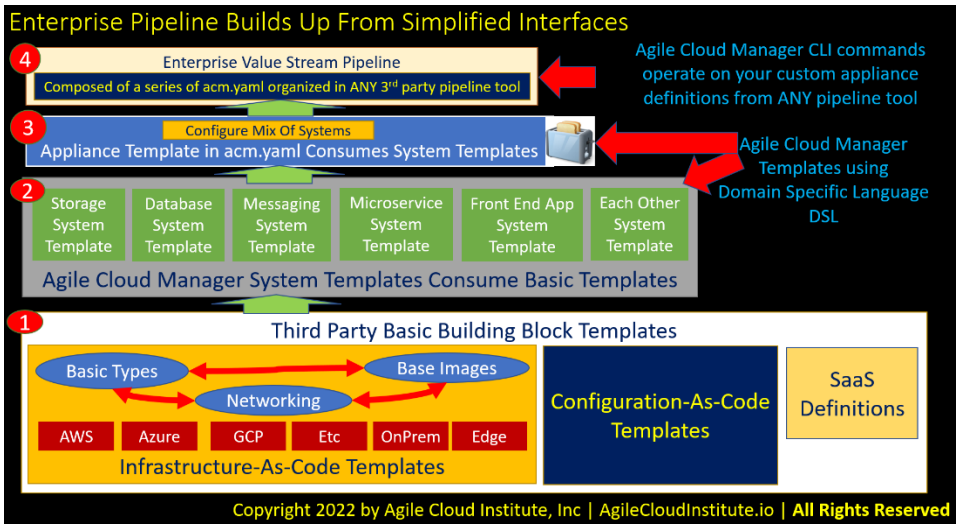


Diagram 6 refers to the Agile Cloud Manager's Domain Specific Language DSL that enables you to model the unique characteristics of your business in terms of custom types of systems and appliances which can then be easily manipulated by pipelines using the Agile Cloud Manager's Command Line Interface CLI.

Your Pre-Existing Basic Building Blocks

At the lowest level, ("1" in the sixth diagram) you can see how your systems can become cloud-agnostic because functionally-comparable versions of basic building blocks can be defined for any public or private cloud.

Infrastructure-as-code templates can define things like networking and basic compute images in ways that are functionally-equivalent between clouds, on-prem, and edges.

The configuration-as-code templates can then be selected as parameters to customize the behavior of the infrastructure building blocks in a way that keeps configuration templates separate from infrastructure templates.

Reusable SaaS definitions can also be placed here.

Everything at this basic building block level of the diagram can be completely abstracted away by the Agile Cloud Manager's cloud-agnostic and vendor-agnostic interface, as you will see in subsequent paragraphs that describe the rest of the diagram.

All your basic building blocks might feed into one single, central repository at the enterprise level via CICD processes for each building block project. Then, for security reasons, each consuming group in your enterprise might only receive access to a specific subset of the building blocks via a subscription to only certain items from the central repository. In this way, you can ensure that each building block is governed, not only from a requirements standpoint, but also from a security standpoint.

System Templates Using Agile Cloud Manager's Domain Specific Language DSL

Next, the system template level, ("2" in the sixth diagram) is where you model your unique business as a collection of basic types of systems, each of which performs specific units of work that have specialized value for your unique business.

Each Agile Cloud Manager system template uses our proprietary Domain Specific Language DSL to define a specialized system such as storage, database, messaging, microservices, front-end apps, or any other type of system in unique terms that meet the unique needs of your business.

All system templates in your enterprise might also feed into one single, central system templates-repository at the enterprise level, following the model described for basic building blocks above. The same system templates might be reused by many groups throughout your enterprise. But each group might only use a small subset of the available system templates that your organization might create.

Rules for publication and subscription of system templates can eliminate redundant engineering tasks while also making sure that each consuming group only receives the specific templates they are going to use. System templates are extremely powerful, so that part of security includes ensuring that each group only has access to the system templates that are actually needed in their own workflows.

Appliances Using Agile Cloud Manager's Domain Specific Language DSL

Third, ("3" in the sixth diagram) is where appliances represent your domains as very simple templates which in turn refer to a small number of custom system templates which each compose meaningful components of your business model.

A brief acm.yaml file summarizes each domain in such simple terms that it can seem like an appliance.

Enterprise Value Stream Pipeline

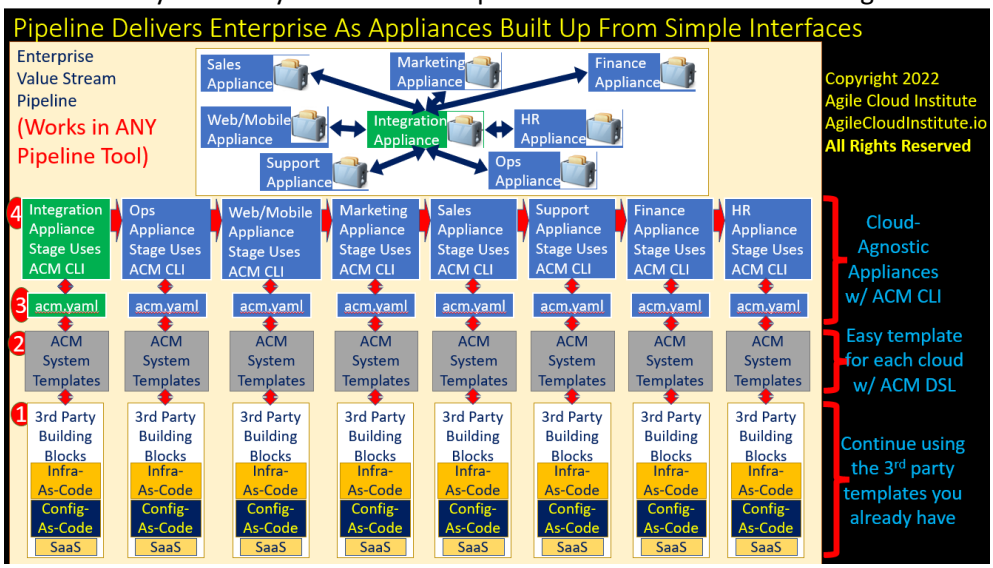
Fourth and finally for the sixth diagram, ("4" in the sixth diagram) your enterprise value stream pipeline is composed of all your appliances, each of which uses its own brief, simple acm.yaml file to represent one of your appliances.

Simple CLI commands enable the pipeline to operate on meaningful aspects of your business because:

- The Agile Cloud Manager's Domain Specific Language DSL enables you to model your systems into unique types.
- And the CLI commands are then able to operate directly on the unique types, which each represent a meaningful aspect of your business.

Each Appliance Has Its Own Stage In Enterprise Pipeline

The seventh diagram, entitled "Pipeline Delivers Enterprise As Appliances Built Up From Simple Interfaces" illustrates how all the systems in your entire enterprise can be delivered in one single pipeline if you use the Agile Cloud Manager to streamline all your systems into reusable appliances.



The top of the seventh diagram revisits the 8 general types of appliances that might appear in enterprise architecture.

The seventh diagram also shows how the 4 layers of each cloud appliance together deliver an appliance in each stage in an enterprise pipeline.

The seventh diagram shows the same 4 levels that diagram 6 showed, but in the seventh diagram you see how each of the 4 levels can appear in its own stage in a pipeline.

Reusable Basic Building Blocks Curated At Enterprise Level

Level 1 at the bottom of the seventh diagram illustrates that you can continue to use all your existing 3rd party templates for infrastructure-as-code, for configuration-as-code, and for SaaS definitions.

Level 1 represents reusable templates which can be curated at the enterprise level, with the same templates available for use in any of the stages, for any appliance, and in any pipeline in your enterprise, subject to rules defined by your enterprise.

Reusable System Templates Curated At Enterprise Level

Level 2 in the seventh diagram shows where system templates written in the Agile Cloud Manager's simple Domain Specific Language DSL organize all your pre-existing lower-level templates into units that have meaningful value to your business.

These custom system templates will make it possible for your higher-level pipelines to operate on units that have business impact on your operations.

In level 2, you can create a separate template in each cloud for each of your systems.

The templates for different clouds will look very similar for the same system because the templates all use the same simple Agile Cloud Manager Domain Specific Language DSL.

The differences between templates in different clouds for the same system will be small things such as slightly different sets of parameters required to interface with different types of lower-level templates for different clouds.

Level 2 also shows re-usable system templates that can be reused in any appliance anywhere in your organization, subject to rules that your enterprise might define.

Cloud-Agnostic Enterprise-Level Interface

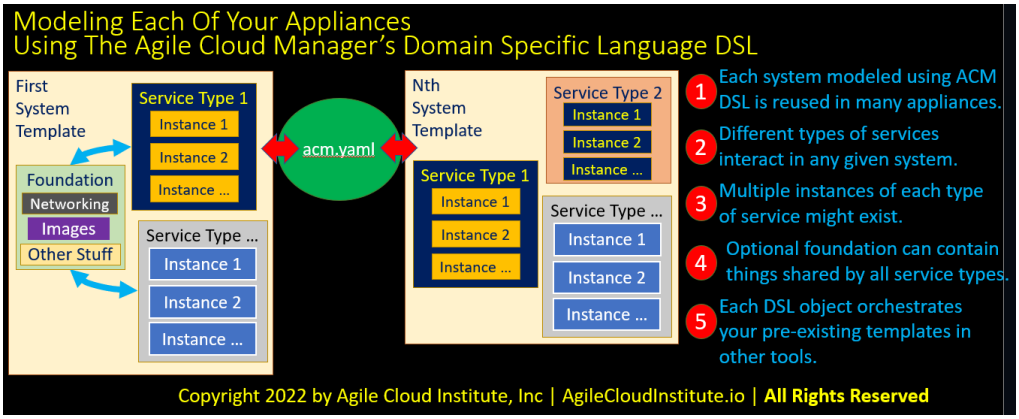
Levels 3 and 4 in the seventh diagram are agnostic with respect to cloud provider because Levels 3 and 4 involve the connection between the Agile Cloud Manager's Command Line Interface CLI and Domain Specific Language DSL.

The CLI is focused solely on your business' unique building blocks which are modeled in the Agile Cloud Manager's Domain Specific Language DSL. The CLI level is therefore completely agnostic with respect to underlying cloud provider. The CLI manipulates business-level objects. And the locations of those business-level objects are outside the scope of the CLI level.

The acm.yaml file that summarizes each of your appliances might contain only a few lines. And each line in each acm.yaml contains only a few simple words. Therefore, changing acm.yaml to switch to a different cloud provider is literally as simple as changing perhaps a couple words on each of perhaps a few lines.

Modeling Each Of Your Appliances Using The Agile Cloud Manager's Domain Specific Language DSL

The eighth diagram illustrates the simple, yet powerful, Domain Specific Language DSL offered by the Agile Cloud Manager.



This DSL enables you to model each of your systems and appliances based on how your unique business operates.

As you can see in diagram 8, each system can be modeled as a collection of some number of types of different services which might interact within the system. Multiple instances of each type of service might exist. An optional foundation can also be

defined to provide items shared by all the different service types, such as networking, images, and other things.

You can continue to use your pre-existing templates from other tools because the Agile Cloud Manager's Domain Specific Language DSL simply orchestrates your pre-existing templates in a way that enables you to deliver appliances more efficiently into enterprise-level pipelines.

Diagram 8 also illustrates how one acm.yaml file summarizes several different systems, which are each defined in their own separate system configuration file.

Diagram 8 is like taking apart a toaster to see that there are a few major subcomponents within each toaster, with each subcomponent abstracting away the complexity of deeper levels.

Diagram 8 also shows how each system configuration can look very much like any other system configuration because the Domain Specific Language DSL offered by the Agile Cloud Manager enables you to organize your systems in a way that is most relevant to your business.

Technical details of the underlying third-party templates and cloud providers are abstracted away by the Agile Cloud Manager's system templates, so that you can organize all your systems in terms of the value that each system provides for your unique business.

Operating On Your Appliances Using The Agile Cloud Manager's Command Line Interface CLI

Twelve operations can be performed using the Command Line Interface CLI offered by the Agile Cloud Manager. Ten of these are business-level operations that can operate surgically on your appliances, and the remaining two operations are for setting up the Agile Cloud Manager.

The Ninth diagram is entitled "Operating On Each Of Your Appliances Using The Agile Cloud Manager's Command Line Interface".

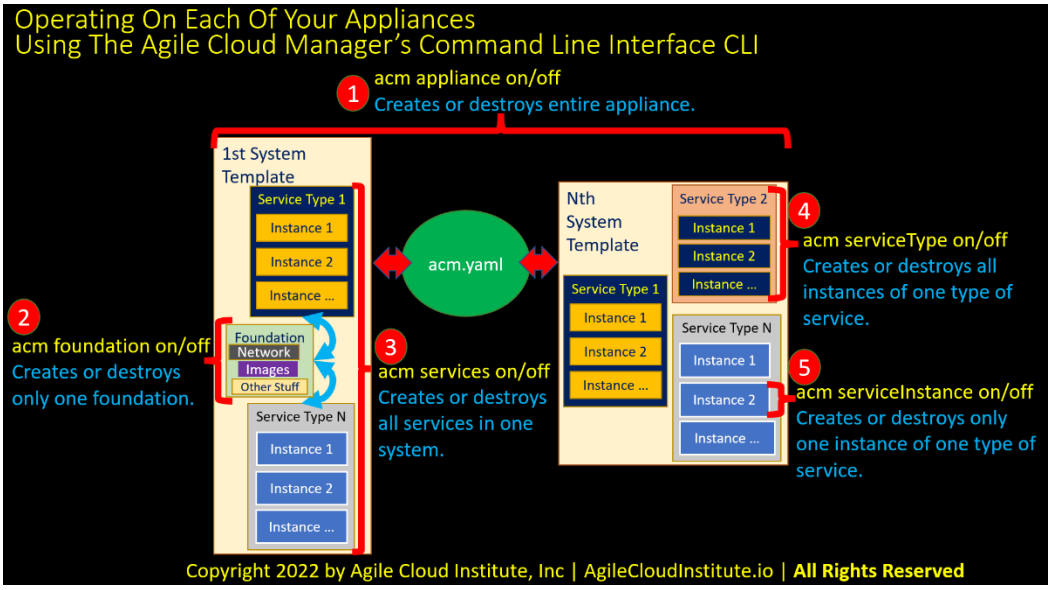


Diagram 9 illustrates how the ten business-level operations are performed by

5 specific pairs of commands which each interact with a specific level of an appliance that is modeled using the Agile Cloud Manager.

The ten business-level operations are on/off switches for 5 levels of the appliances and systems that you have seen in the diagrams above so far in this article. These 5 levels and 10 commands are referenced by numbers 1 through 5 in the diagram and are:

Number 1 in the diagram is “acm appliance on/off”, which you can see operates on the entire appliance.

- `acm appliance on` installs the entire appliance in one single command
- `acm appliance off` destroys the entire appliance in one single command

Number 2 in the diagram is “acm foundation on/off”, which operates on one foundation.

- `acm foundation on <flags>` creates only the foundation of only one system
- `acm foundation off <flags>` destroys only the foundation of only one system

Number 3 in the diagram is “acm services on/off”, which operates on all the services in one system.

- `acm services on <flags>` creates all the services for only one system. This means creating all the instances of every type of service defined in the system configuration.
- `acm services off <flags>` destroys all the services for only one system. This means destroying every instance of every type of service defined in the system configuration.

Number 4 in the diagram is “acm serviceType on/off”, which operates on only one type of service.

- `acm serviceType on <flags>` creates all the instances of only one type of service for only one system.
- `acm serviceType off <flags>` destroys all the instances of only one type of service for only one system.

Number 5 in the diagram is “acm serviceInstance on/off”, which operates on only 1 instance of a service.

- `acm serviceInstance on <flags>` creates only one specific instance of only one specific type of service in only one system.
- `acm serviceInstance off <flags>` destroys only one specific instance of only one specific type of service in only one system.

In addition, and not visible in the diagram, there is “acm setup on/off” which together enable the Agile Cloud Manager to work from any sufficiently provisioned Windows or Linux computer that is sufficiently connected to the internet.

- `acm setup on <flags>` prepares a computer to operate on an appliance. This includes retrieving all the various repositories of templates that will be orchestrated into the specific appliance. But note you must provision the compute instance properly before running `acm setup on <flags>`.
- `acm setup off` removes all the things that the `acm setup on` downloaded

These 12 commands together give you the ability to operate on your business at a high-level.

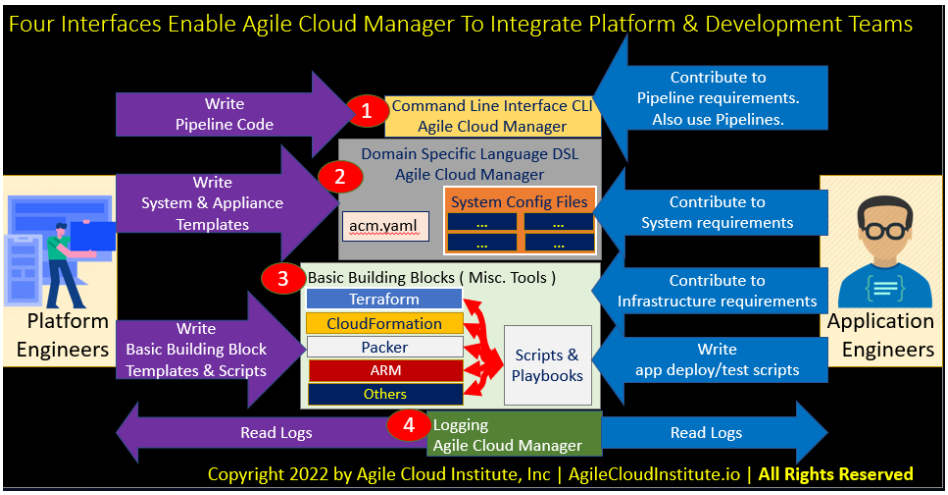
These 12 commands enable you to abstract away huge amounts of technical complexity, so that it becomes easier to drive change in your business.

These 12 commands enable you to reduce the cost of your IT organization because everything below these 12 commands can be organized into reusable templates that can be:

- Put under Agile project management.
- Governed by your enterprise governance systems.

Four Interfaces Enable The Agile Cloud Manager To Integrate Platform & Development Teams

The tenth diagram illustrates the separation of concerns between your platform engineers and your application engineers that becomes possible when you use the Agile Cloud Manager.



The tenth diagram illustrates the 4 types of interfaces that are exposed by the Agile Cloud Manager, numbered 1 through 4 in red.

Application engineer roles are on the right side of the diagram, while platform engineer roles are on the left side of the diagram.

You can see in diagram 10 that application engineers only need to write narrowly-scoped deployment and test scripts related to their own application. More details of application-level deployments are given in the next section.

Similarly, platform engineers write appliance code, which is composed of:

- Reusable basic building blocks
- Reusable system templates
- Brief appliance definitions that reuse the system templates, and
- Pipeline scripts that use the Agile Cloud Manager's Command Line Interface CLI

Both application engineers and platform engineers alike each read the logs produced by the Agile Cloud Manager. Logging is explained in the engineering-level documentation on this website.

The tenth diagram addresses the work of writing code and does not speak to the ultimate accountability for requirements which will be determined by your organization's own unique governance processes.

Remember that the same templates for each system and for each basic building block will be reused by many different appliances throughout your enterprise.

Therefore, the reusability of modules at every level with the Agile Cloud Manager elevates the design of reusable modules to a more centralized level in your organization.

Requirements will therefore be defined by numerous different types of stakeholders, so that Diagram 10 illustrates that application engineers contribute to requirements definitions.

How Meta-Configuration Enables Re-Use Of Agile Cloud Manager Templates

Two very simple external files named keys.yaml and config.yaml can enable all of your system templates and appliance templates to become widely reusable across your organization. Both keys.yaml and config.yaml have a very simple structure composed of a list of key/value pairs.

The purpose of keys.yaml and of config.yaml is simply to store the data that enable the same system and appliance templates to be cloned as many times as needed.

keys.yaml is a file that the Agile Cloud Manager uses to store sensitive secrets that allow authentication with remote systems. Your automation can secure keys.yaml in each agent first by sourcing its contents from a secrets vault, then by locking down the directory in which it is placed in each agent, and finally by destroying it from each agent after it is no longer needed.

config.yaml has the same simple structure as keys.yaml. But the difference is that config.yaml is intended to store non-secret values which enable the Agile Cloud Manager to use unique names to create cloud resources whenever a clone of a system needs to be created.

The values stored in keys.yaml and in config.yaml can be ingested into Agile Cloud Manager system templates at runtime using a very simple variable-mapping syntax that is described in the engineering-level documentation in this web site.

How Custom Controllers Allow You To Extend The Agile Cloud Manager

The Agile Cloud Manager ships with many pre-installed controllers that enable it to operate on a variety of third-party automation tools including Cloud Formation, ARM Templates, Terraform Modules, Packer Templates, and others.

However, you can also create your own custom controllers which you can use to extend the Agile Cloud Manager to meet specialized requirements within your enterprise.

The eleventh diagram is entitled “Custom Controllers Can Extend The Agile Cloud Manager”.

Diagram 11 illustrates how a custom controller that you create can act as a go-between to connect the Agile Cloud Manager’s automation algorithms with any third-tool with which you might want to integrate.

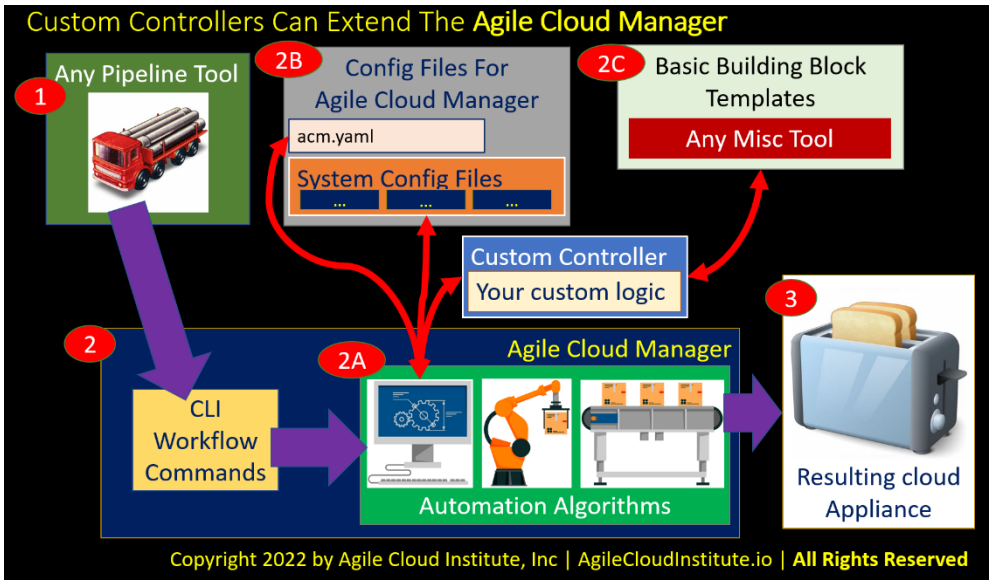


Diagram 11 illustrates how your own custom controllers can be orchestrated using the same Agile Cloud Manager CLI commands and the same system templates and appliance templates written in the same Agile Cloud Manager Domain Specific Language DSL.

The difference is that your custom controllers can do anything that you program them to do.

This means that custom controllers that you develop can enable your extensive preexisting business logic to become more fully integrated into the enterprise-level pipelines that the Agile Cloud Manager makes possible.

A working example of a custom controller is available in our GitHub site and is described in the engineering section of this web site.

Custom controllers can potentially be written in any programming language. But the working example that we provide is of a custom controller written in Python.

If you decide that you require a custom controller, we recommend that you begin by running our working example of a custom controller, and that you then incrementally experiment with adding new functionality to it.

Multi-Application Deployments With The Agile Cloud Manager

Multiple applications need to be updated at the same time in an Agile enterprise because customer-centric workflows are the focus of an Agile organization.

The twelfth diagram is entitled "Planning & Deploying Customer-Centered Enterprise Changes".

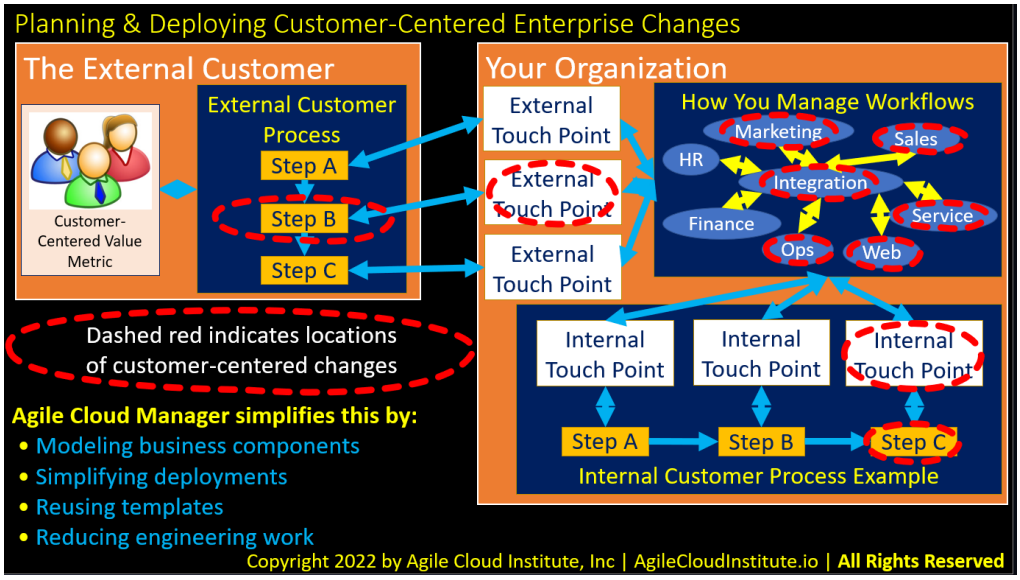


Diagram 12 illustrates how improving a workflow usually means making small changes to several applications because workflows span multiple systems throughout an organization.

The Agile Cloud Manager greatly simplifies multi-application deployments.

Agile requirements-gathering and Agile architecture require an enterprise-level view of all the appliances that can be modeled by the Agile Cloud Manager.

In fact, the Agile Cloud Manager was designed specifically to make it a lot easier to focus engineering on much faster delivery of upgrades to enterprise-level customer-defined value.

The delivery pipeline for the kind of multi-application deployment illustrated in Diagram 12 can look a lot like the enterprise pipeline illustrated in previous diagrams earlier in this document.

The difference is that a multi-application deployment like what we see in Diagram 12 will involve some applications changing while other applications remain the same.

Multi-application deployments can be done as either Blue/Green deployments or as Canary deployments.

For a Blue/Green deployment using the Agile Cloud Manager, one option is to simply use the same exact enterprise pipeline described above, except to use the newly upgraded app artifacts at each stage as inputs into the same pipeline.

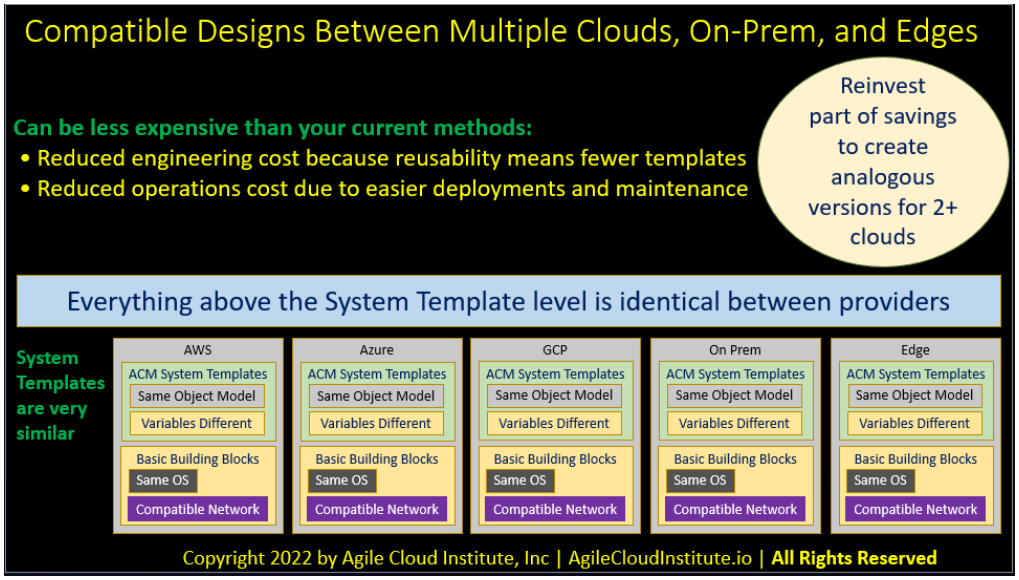
For a Canary deployment using the Agile Cloud Manager, you might use a broader range of the CLI commands offered by the Agile Cloud Manager instead of simply `acm appliance on` and `acm appliance off`.

Some of the various deployment-type options are described in more detail in our separate article "Multi-Application Deployments With The Agile Cloud Manager".

Fail-Over and Fail-Back Between Any Cloud, On-Prem, or Edge

The two diagrams in this section illustrate how designing each of the requirements for failover and failback can become a very simple process using the Agile Cloud Manager.

The Thirteenth diagram illustrates some of the aspects of designing for multiple clouds, on-prem, and edges using the Agile Cloud Manager.



A budget for multi-cloud can be created from savings that can be accrued from the efficiencies that result from the reduced engineering and operations costs that can result from using the Agile Cloud Manager.

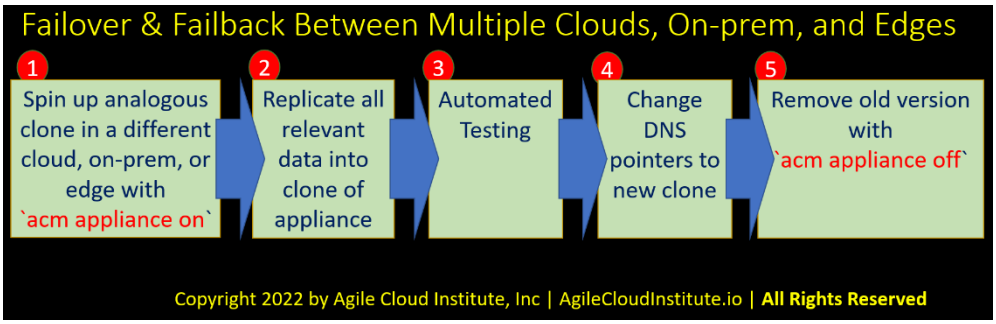
Everything above the system template level can be virtually identical between cloud providers due to the Agile Cloud Manager's Command Line Interface CLI's ability to abstract away the inner details of systems.

System templates for the same system in different clouds will be almost identical because the Agile Cloud Manager's Domain Specific Language DSL enables you to model the same system the same way for every cloud, with the same service types, the same service instances, and the same foundation.

The only thing that might be different between system templates for the same system in different clouds might be the variables needing to be mapped differently within each object to fit the underlying building blocks.

Basic building blocks for different clouds can be analogous to each other if you are careful to use identical operating systems and compatible network designs.

The fourteenth diagram illustrates how simple pipeline design can be for Failover and Failback using the Agile Cloud Manager.



Failover and Failback between cloud, on-prem, and edge providers is a lot like doing Blue/Green deployments in the sense that your automation creates a complete cloned copy of your Production environment and then redirects traffic to the newly cloned Production environment. The differences are that:

- The new clone of Production is in a different cloud, different on-prem, or different edge.
- The application versions in the new Production are the same before and after a failover or a failback.

Data replication between the cloned appliances becomes a much more isolated and reasonably scoped engineering task.

Governance, Security, and Compliance

The fifteenth diagram is entitled "Enterprise Governance Simplified By Agile Cloud Manager".

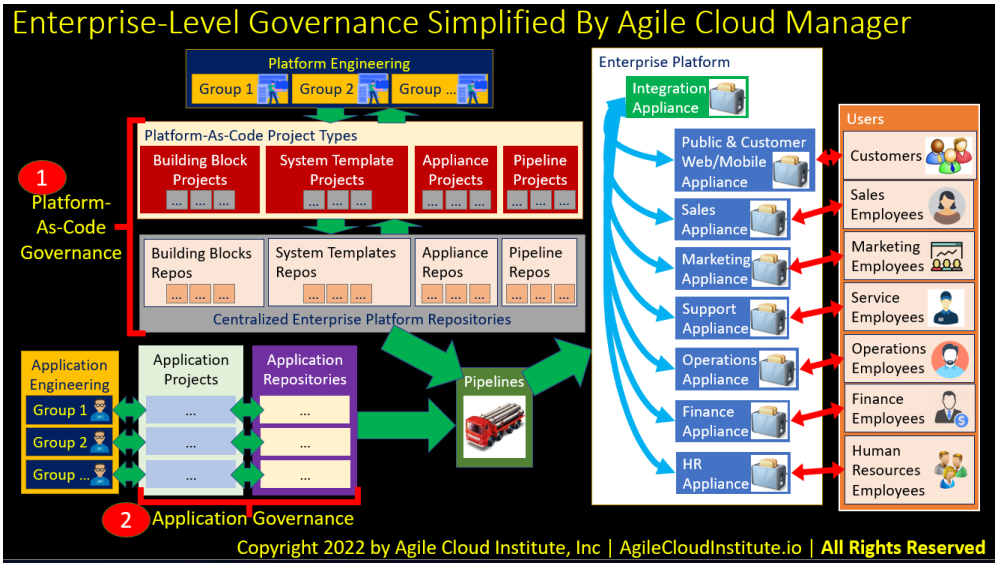


Diagram 15 illustrates how the discretely organized components and the reliance on pipelines together make it very easy to put each element of completely software-defined platforms into a governance process.

In diagram 15, you can clearly see how "Platform-As-Code" can be decomposed into at least 4 categories of Agile projects, and that each Agile project can manage specific code repositories, which together comprise an entire software-defined enterprise platform.

This can also reduce the size of your enterprise attack surface because the same reusable templates constitute a smaller amount of code doing the same things. You can focus on improving the security of the smaller amount of code that composes your smaller attack surface.

And compliance can become easier because a smaller number of items need to be reviewed, and because each item has clear accountability to the bigger picture.

A more detailed discussion of governance is given in our articles "Architectural Migration and Governance" and "Quantifying the Financial Benefits".

Review How Each Use Case Is Fulfilled

Now that we have discussed the main architectural aspects, the following table will show how each of the defined use cases is addressed by specific aspects of the Agile Cloud Manager.

Use Cases	Elements that fulfill Use Case
Upgrade multiple applications in parallel to accelerate customer value creation, increasing the speed with which new features can be implemented	Appliances and reusable system templates with Agile Cloud Manager's DSL controlled efficiently by the Agile Cloud Manager's CLI. Smaller number of reusable basic building blocks. Simplified application-level deployment experience. Enterprise-level pipelines.
Failover and Failback between different cloud providers	ACM CLI enables pipelines to run the same command regardless of the underlying infrastructure. Vendor-specific implementations of each system-template and of each basic building block created for each cloud vendor that you want to support.
Reduce redundant engineering efforts	Reusable system templates, reusable basic building blocks, and standardized pipelines.
Encapsulate security components	Reusable system templates, reusable basic building blocks, and standardized pipelines
Create cloned environments quickly	Change values in keys.yaml and in config.yaml
Manage business-level components more easily	Custom types of systems and of services enable your code to be encapsulated within units that have meaning for your business. Custom controllers that you can create can enable specialized behavior. CLI commands enable you to perform operations on your custom system types, custom service types, and individual service instances.
Integrate all engineering work into a single enterprise pipeline	CLI and DSL enable operations to be performed on units that have specific meaning to the business. Reusable system configurations and building blocks mean that differences between

	implementations are mainly in simple configuration so that less work is required to prove out new ideas.
Put infrastructure-as-code and configuration-as-code under agile project management	Software definition of every element with architectural diagrams makes it easy to integrate with project management tools. Enterprise-level pipelines make every element accountable for project management.
Simplify governance, security, and compliance	Rules are baked-into reusable building blocks. Pipelines control the use of each building block. Everything is modeled as a custom type relevant to your unique business.
Integrate any pipeline tool with infrastructure-as-code and with configuration-as-code	Simple CLI moves complexity out of pipeline scripts and into portable configuration files that can be used with any pipeline tool.
Reduce the amount that engineers are required to learn	Interfaces and code separation isolate scope of every engineering task.

For more information, you can contact us at http://AgileCloudInstitute.io/contact

