Duck Creek Technologies

Duck Creek Architecture

What is changing?

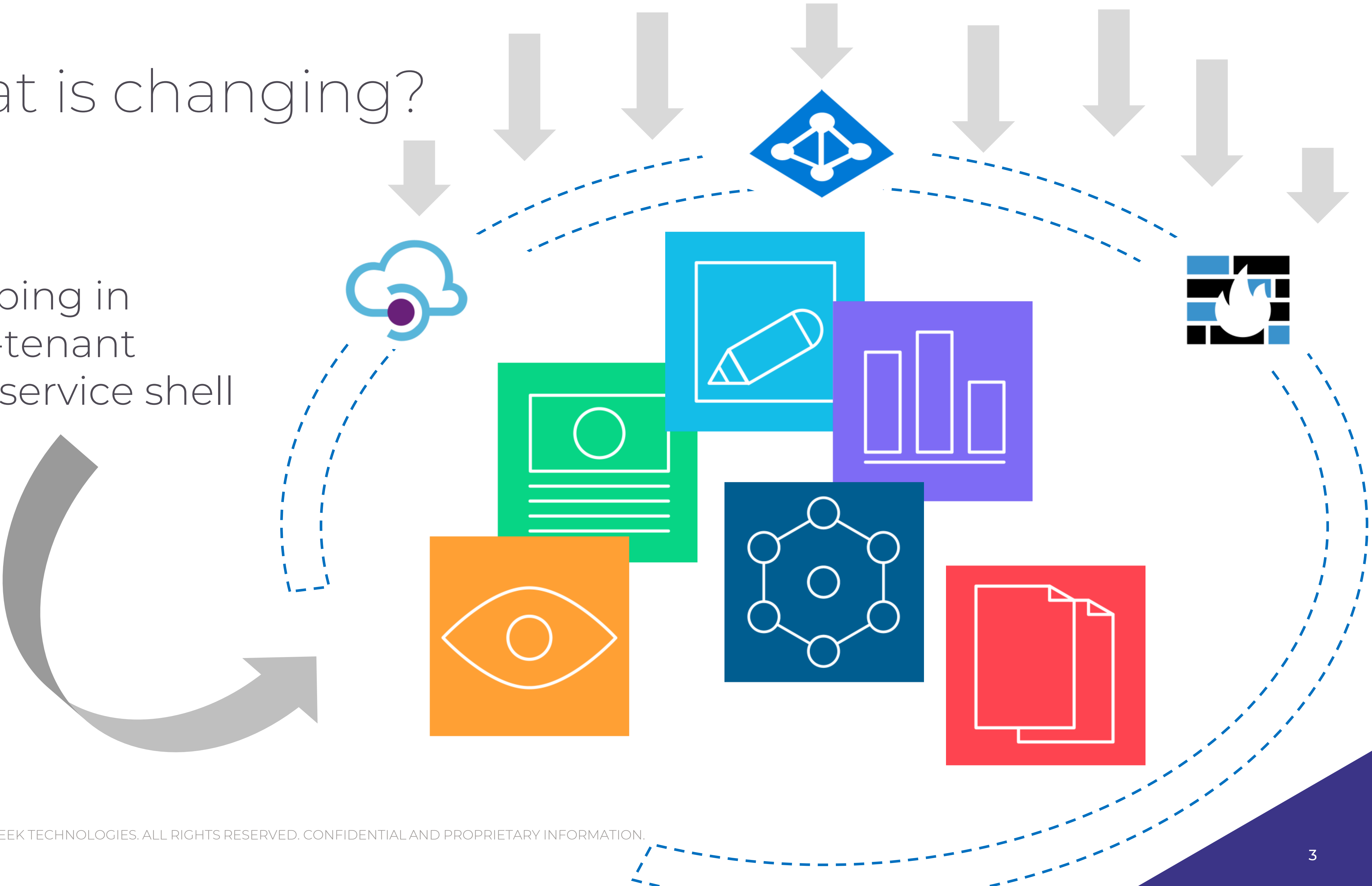
Deployment Architecture is the design and architecture of the platform on which the coded software will actually be deployed when it is running in a production environment or test environment.

The current release is all about deployment architecture.

*Moving from **supported by** the Azure platform to running natively **on** the Azure platform*

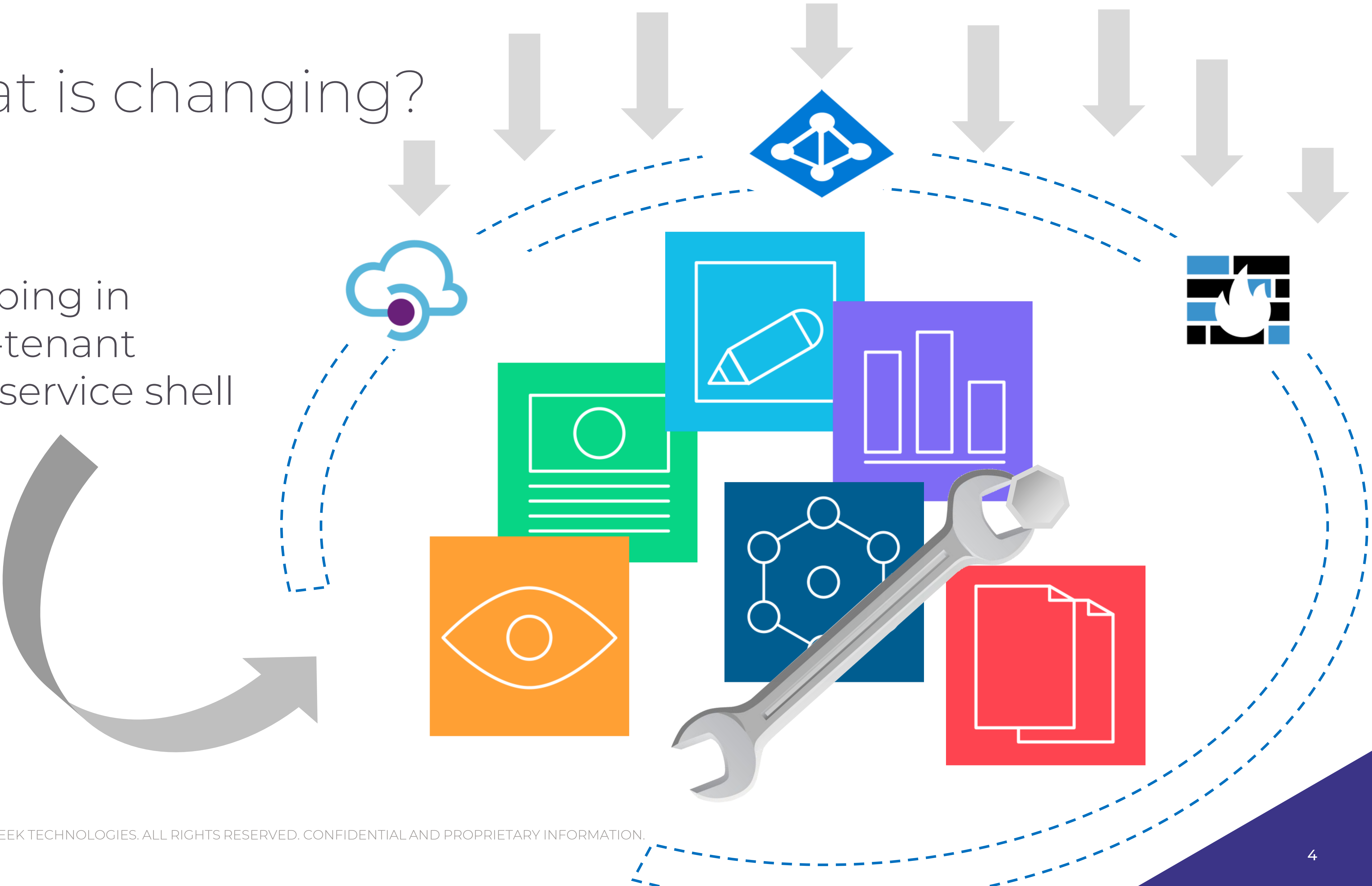
What is changing?

Wrapping in multi-tenant microservice shell



What is changing?

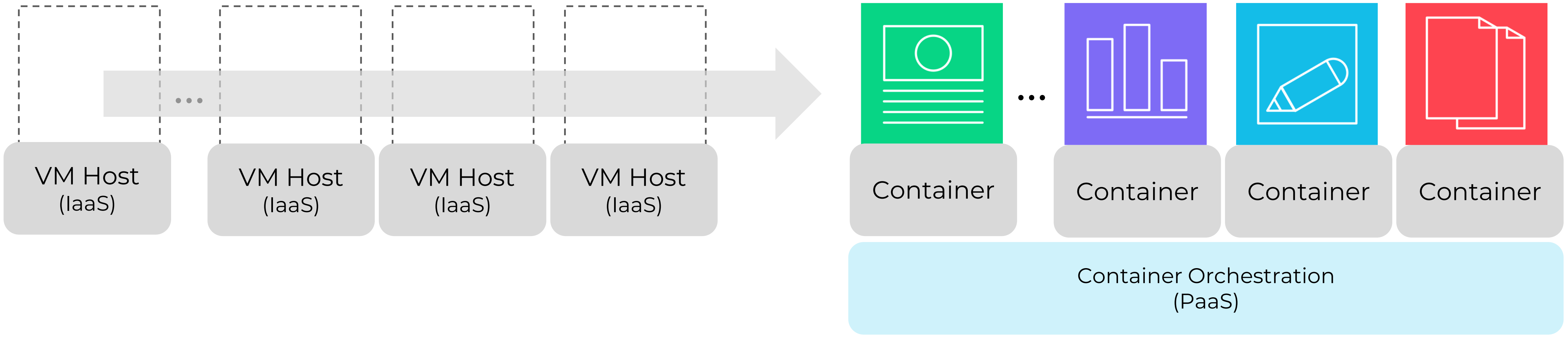
Wrapping in multi-tenant microservice shell



What is changing?

#1

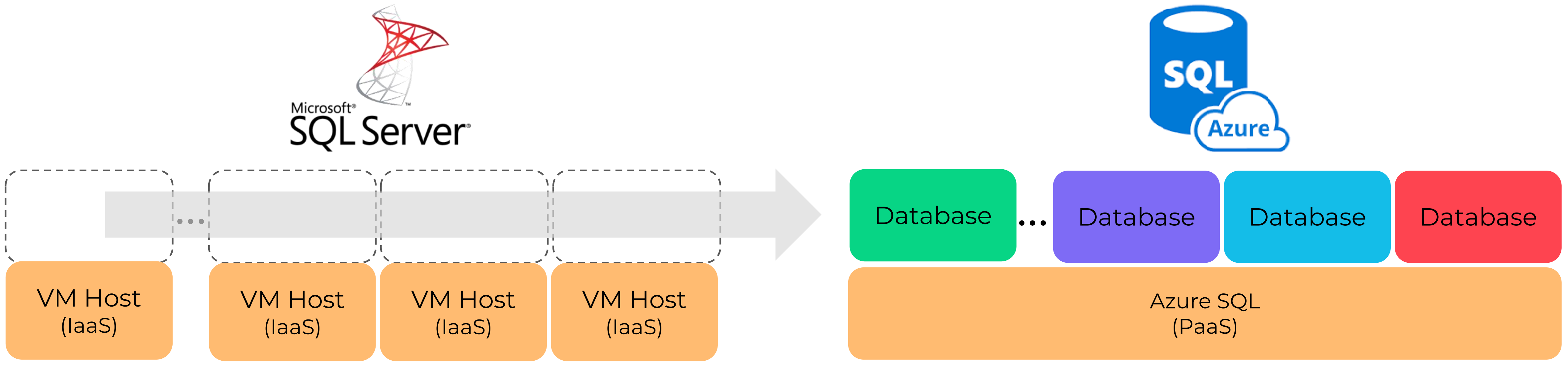
Moving from IaaS application hosts to PaaS hosts



What is changing?

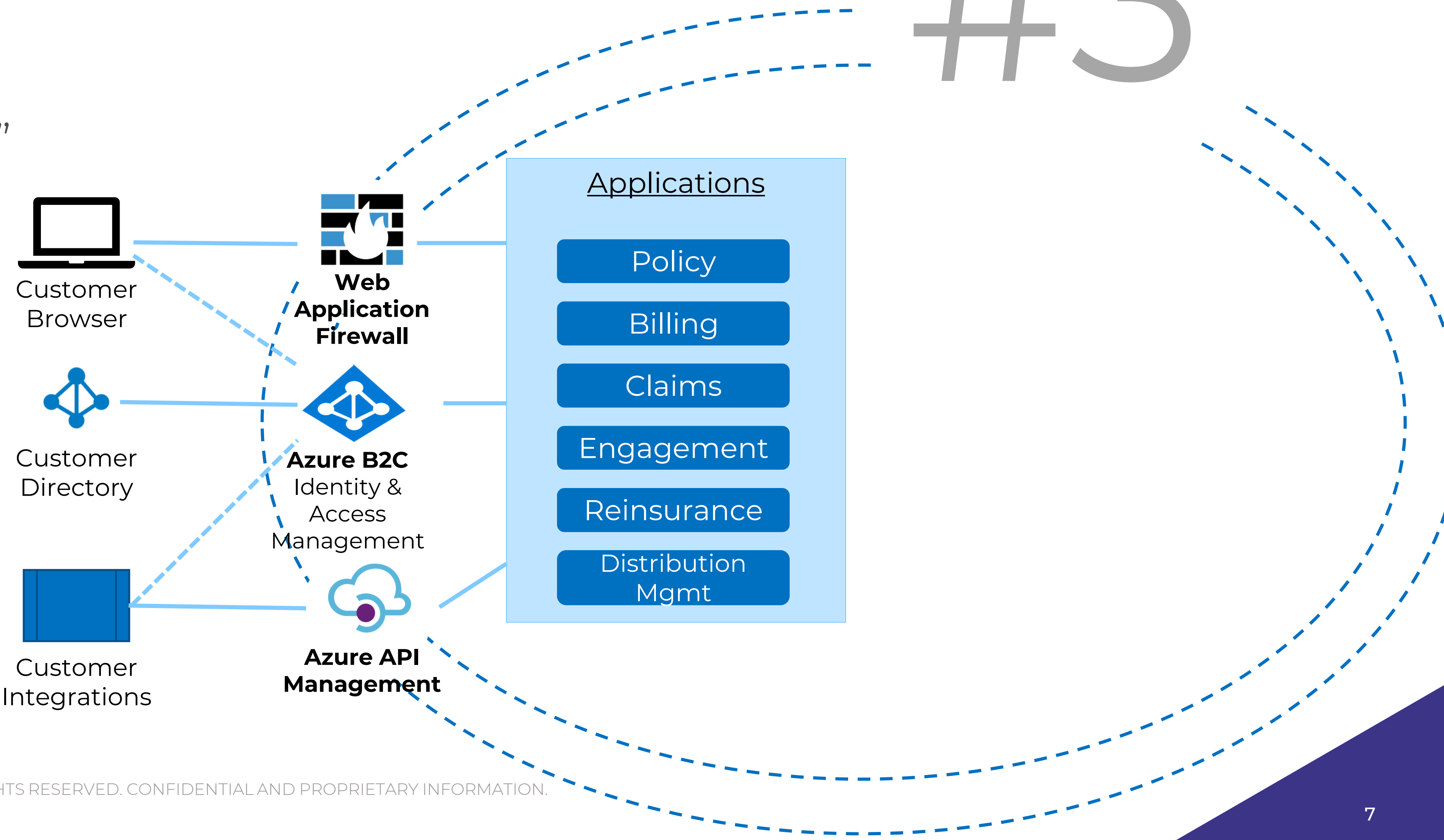
#2

Moving from databases on VM hosted SQL Server instances to Azure SQL databases



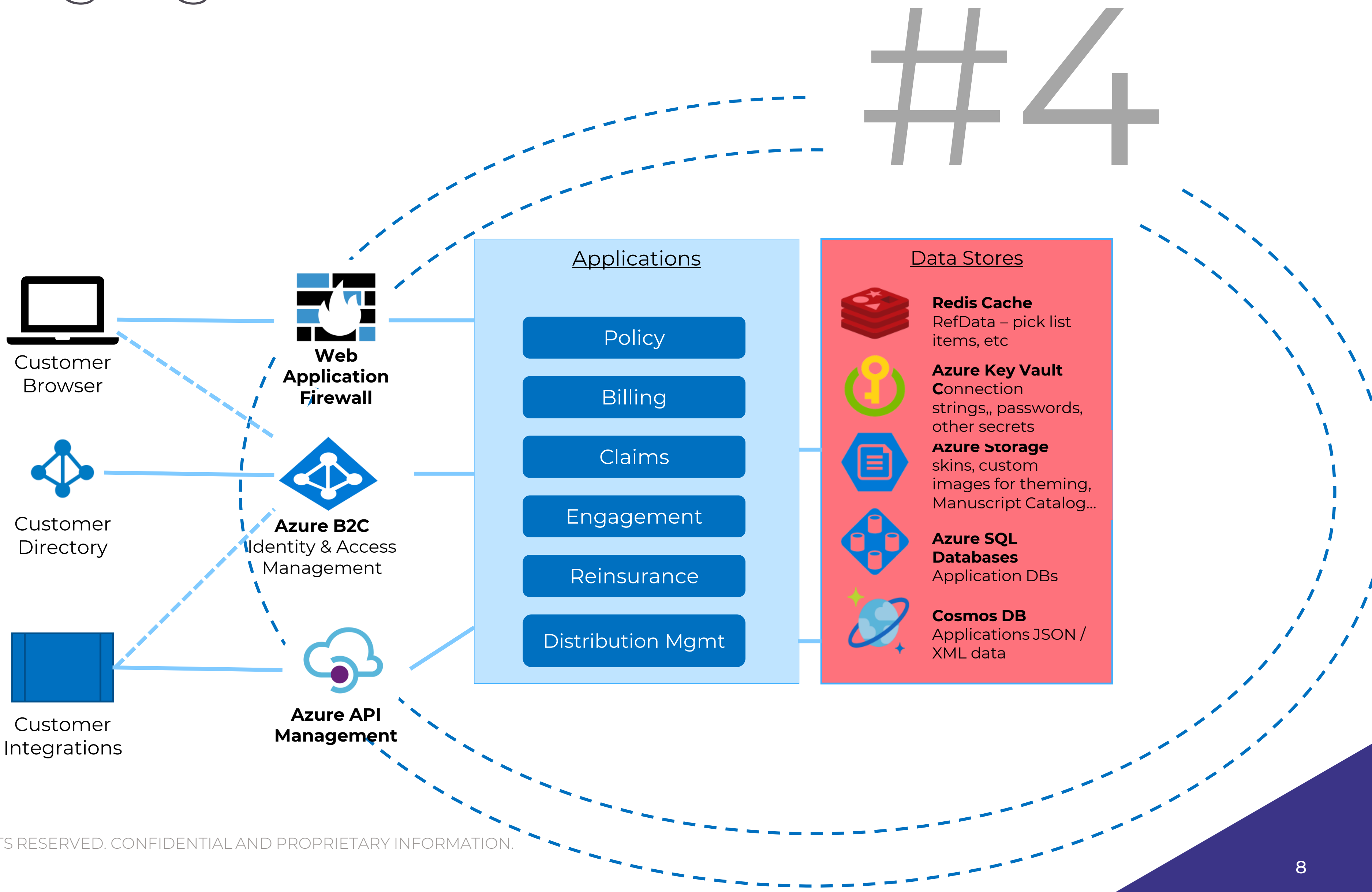
What is changing?

New "front door" into the applications



What is changing?

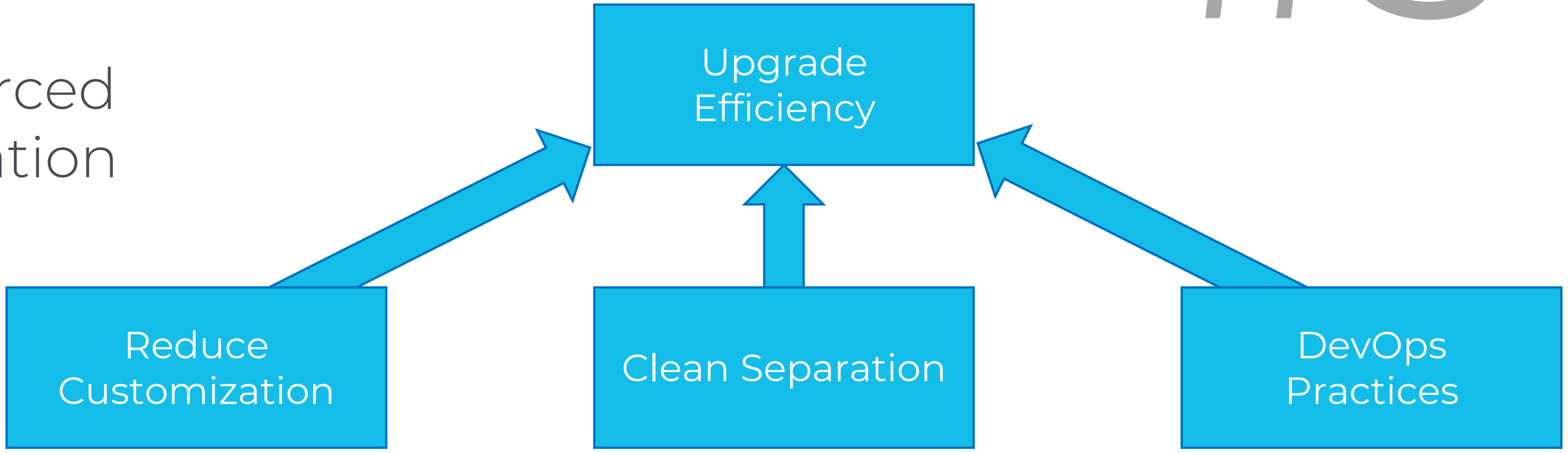
Fit-for-purpose storage


#4
Customer Browser
Customer Directory
Customer Integrations
Web Application Firewall
Azure B2C Identity & Access Management
Azure API Management
Applications
Policy
Billing
Claims
Engagement
Reinsurance
Distribution Mgmt
Data Stores
Redis Cache RefData – pick list items, etc
Azure Key Vault Connection strings,, passwords, other secrets
Azure Storage skins, custom images for theming, Manuscript Catalog...
Azure SQL Databases Application DBs
Cosmos DB Applications JSON / XML data

What is changing?

#5

Strictly enforced implementation patterns



Where possible, replace customization with configuration

Do not mix client code/configuration with Duck code/configuration

Automated Testing,
Automated Deployment
Telemetry
Change Management

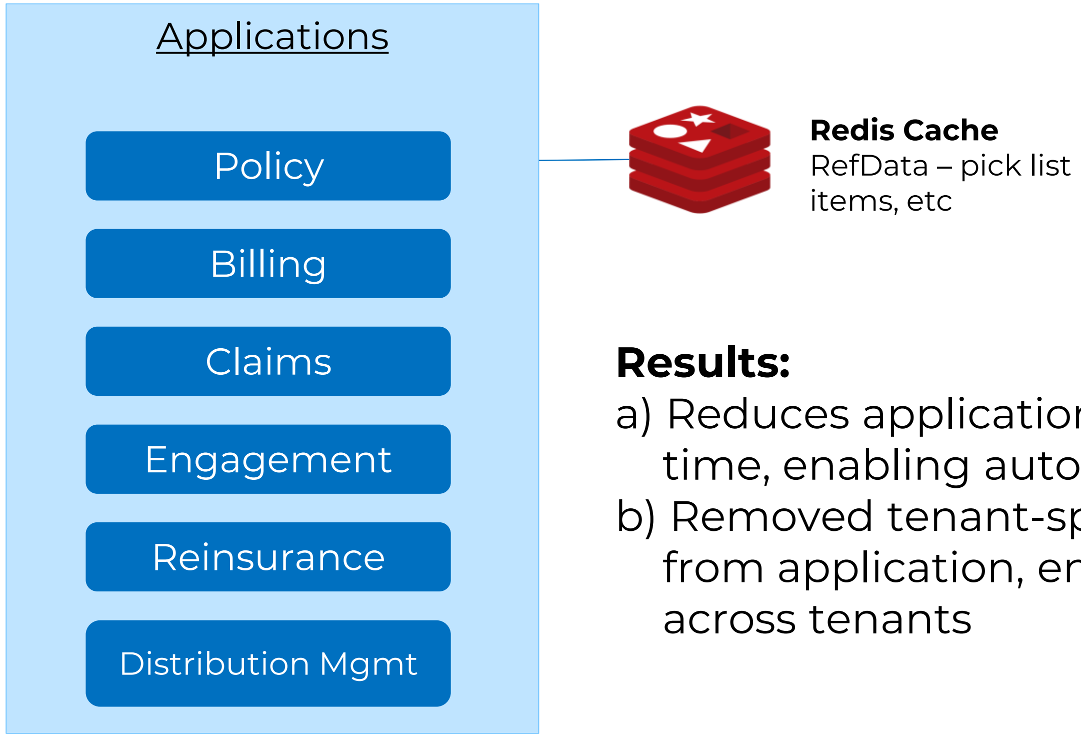
"Right Way"

"Right Place"

"Right Process"

Redis Cache

What:
Redis is an in-memory data store
Use:
Used to cache reference data – i.e., pick lists, UI strings, etc.
Why:
This data changes infrequently; performance serving from cache is much better than round-tripping from DB
Change:
In the past, this data was cached by the application itself. This had 2 problems: a) it was tenant-specific, locking an application instance to a single tenant, and b) it took a long time to warm-up, which made auto-scaling impractical



Results:
a) Reduces application warm-up time, enabling auto-scaling
b) Removed tenant-specific data from application, enabling sharing across tenants

Azure Key Vault

What:

Key Vault is a multi-tenant cloud service used to store and manage keys, secrets and certificates

Use:

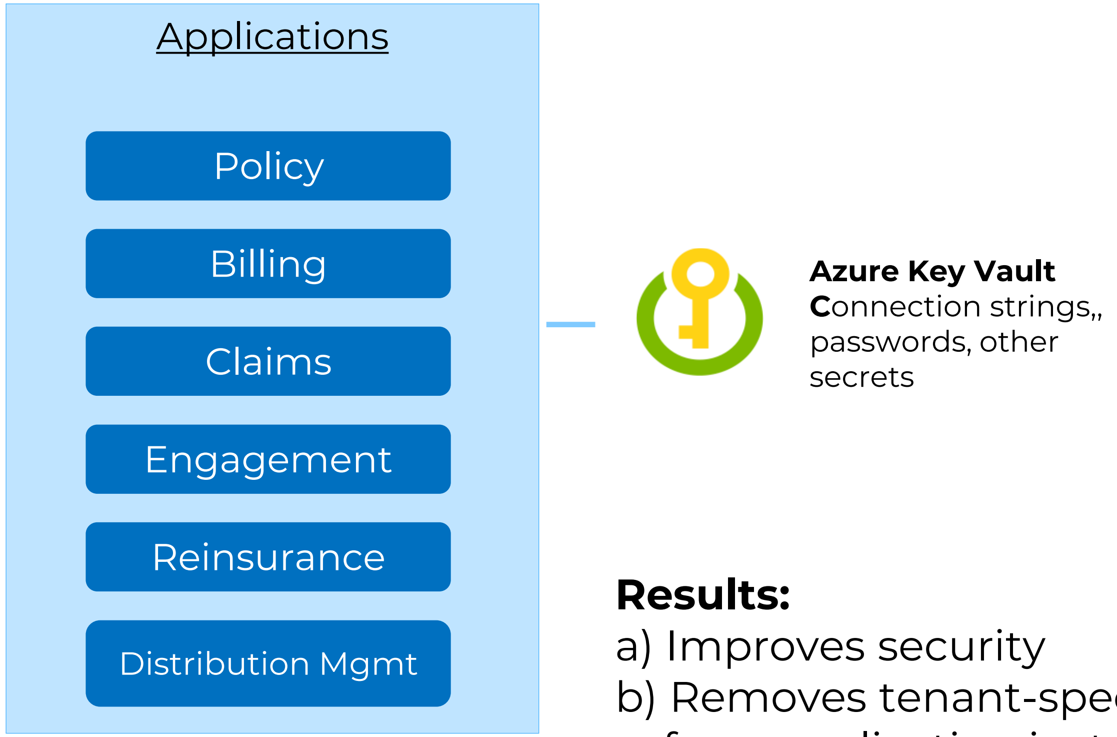
Store and manage per-tenant keys and secrets

Why:

Secure management of this information, separated from the application

Change:

In the past, this data was typically stored in encrypted configuration files local to the consuming application. Moving this out of the application removes tenant-specific information from the application and improves the security of this application.



Results:

a) Improves security
b) Removes tenant-specific data from application instance
c) Enabler for architecting the applications with a key-centric authorization approach

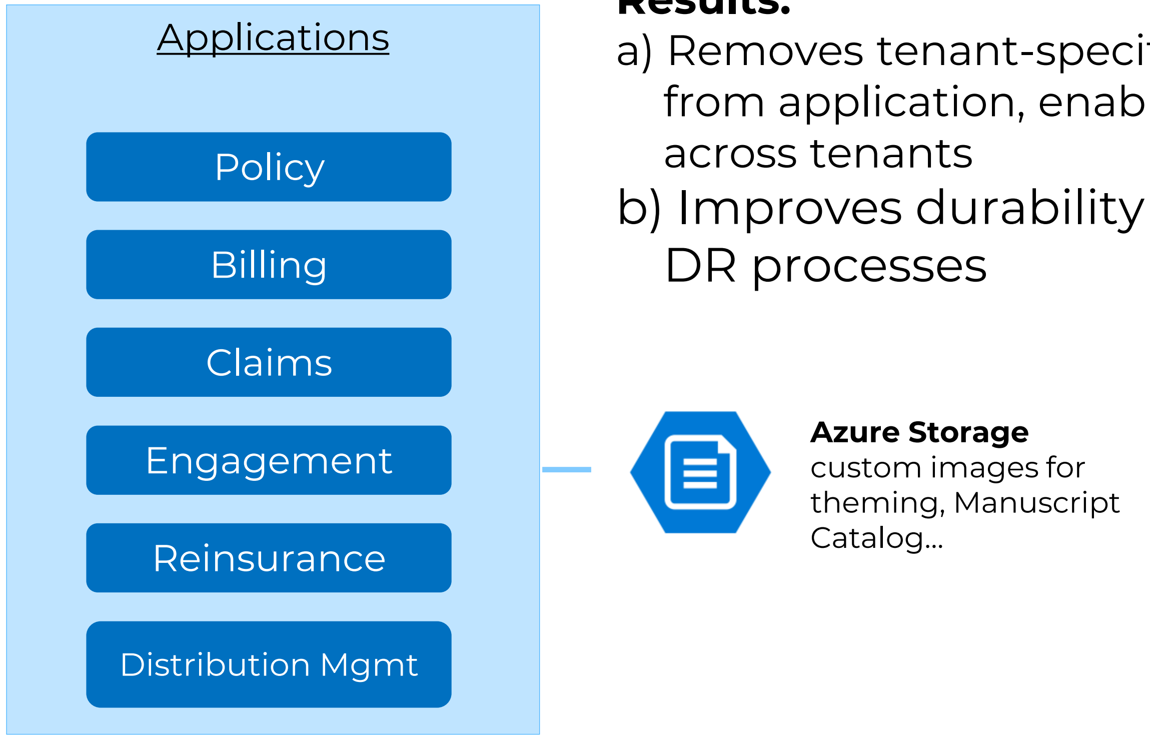
Azure Storage

What:
Secure, scalable cloud-based storage

Use:
File storage

Why:
Applications use file storage for things like the Manuscript catalog, custom images for theming, configuration files, etc.

Change:
In the past, this data was stored in the local files system of the application instance. Moving this to cloud storage removes tenant-specific information from the application instance and improves durability / backup / DR processes



Results:
a) Removes tenant-specific data from application, enabling sharing across tenants
b) Improves durability / backup / DR processes

Azure SQL

What:

Azure SQL Database is a general-purpose relational database, provided as a managed service.

Use:

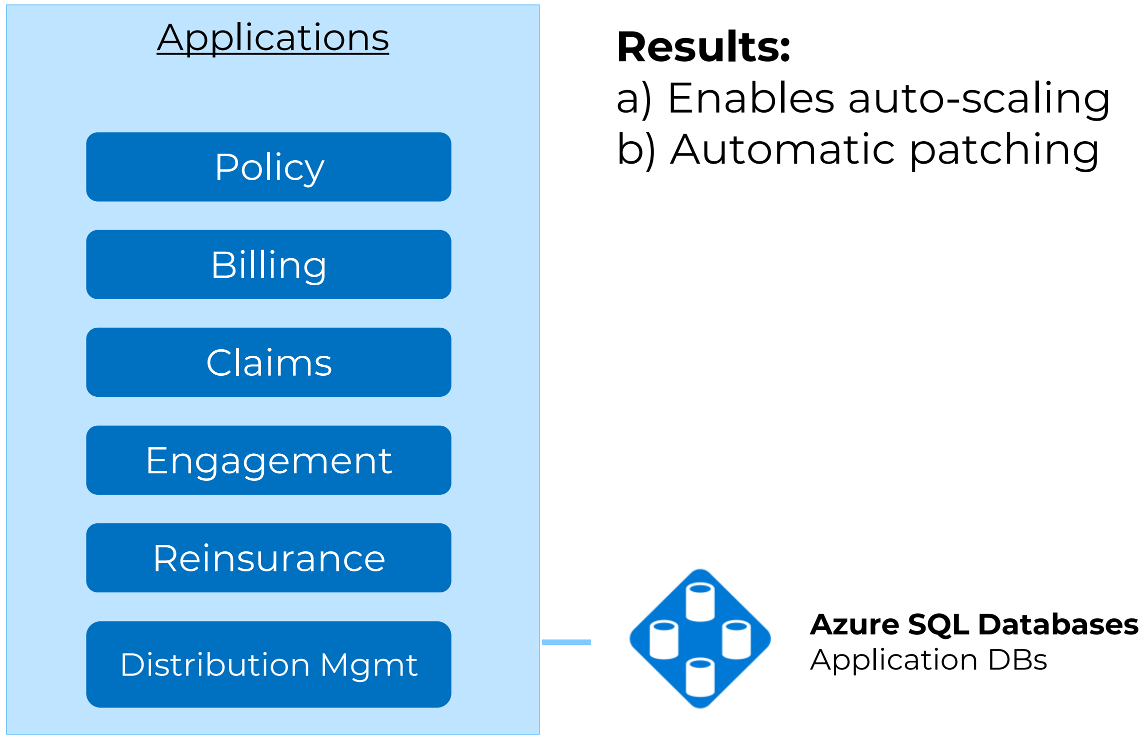
Persistence of DC application relational data – both transactional and configuration

Why:

More flexible scalability

Change:

In the past, this data was stored in SQL Database on per-tenant dedicated virtual machines. Scaling that required provisioning new VMs. With Azure SQL and resource pools, capacity can be automatically scaled across tenants.



Results:

a) Enables auto-scaling
b) Automatic patching

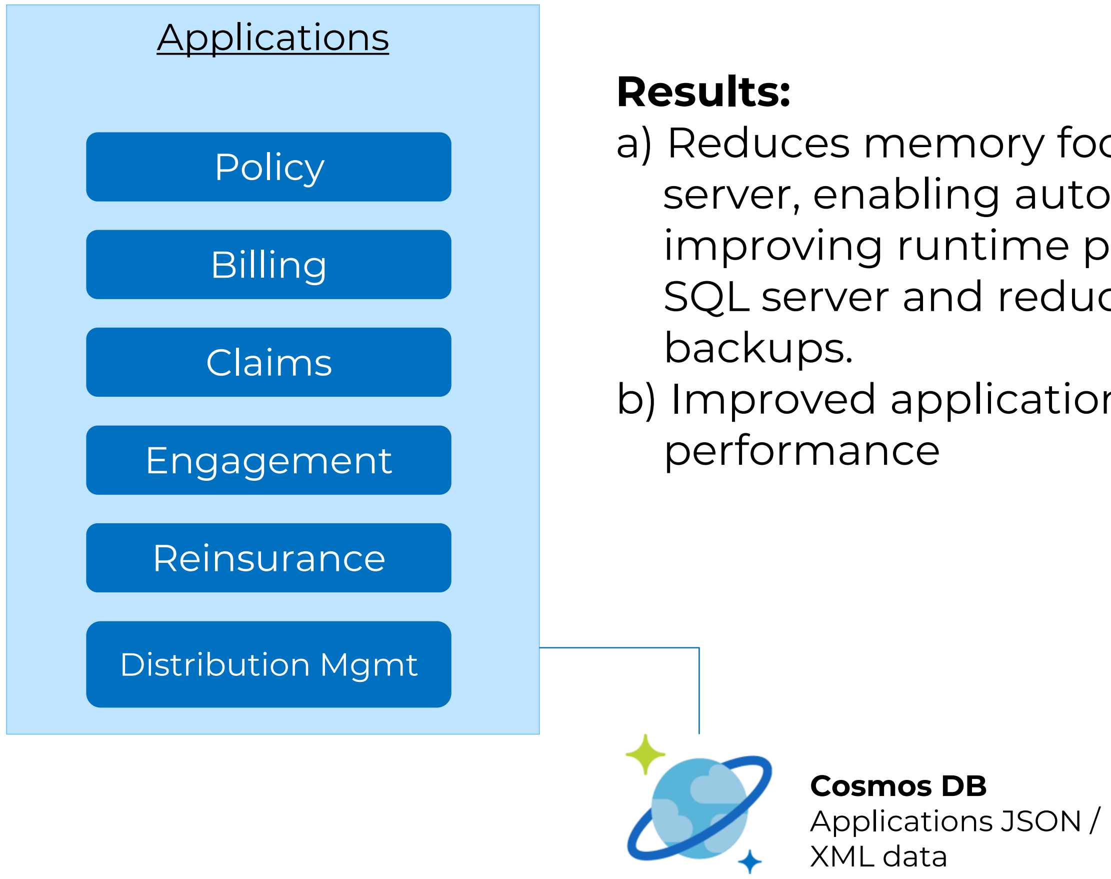
Cosmos DB

What:
CosmosDB is a cloud-based NoSQL DB

Use:
Persist session data

Why:
Provides superior query performance for non-relational data vs. SQL server; off-loads data from SQL server.

Change:
In the past, this data was persisted in SQL server.



Results:

a) Reduces memory footprint of SQL server, enabling auto-scaling, improving runtime performance of SQL server and reducing time for backups.

b) Improved application performance

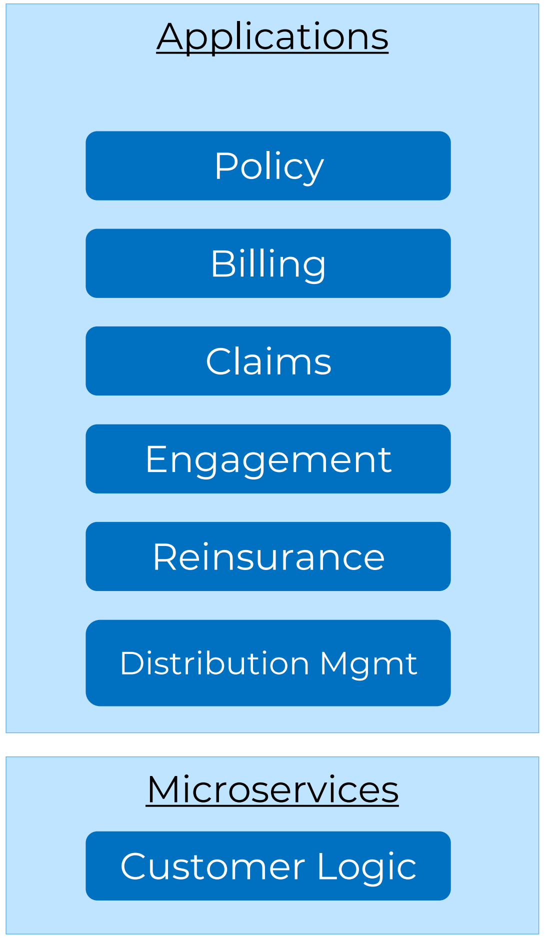
Microservice Architecture

What:
Customer logic hosted as stand-alone, API-based microservices

Use:
Allows customers to extend the DC applications

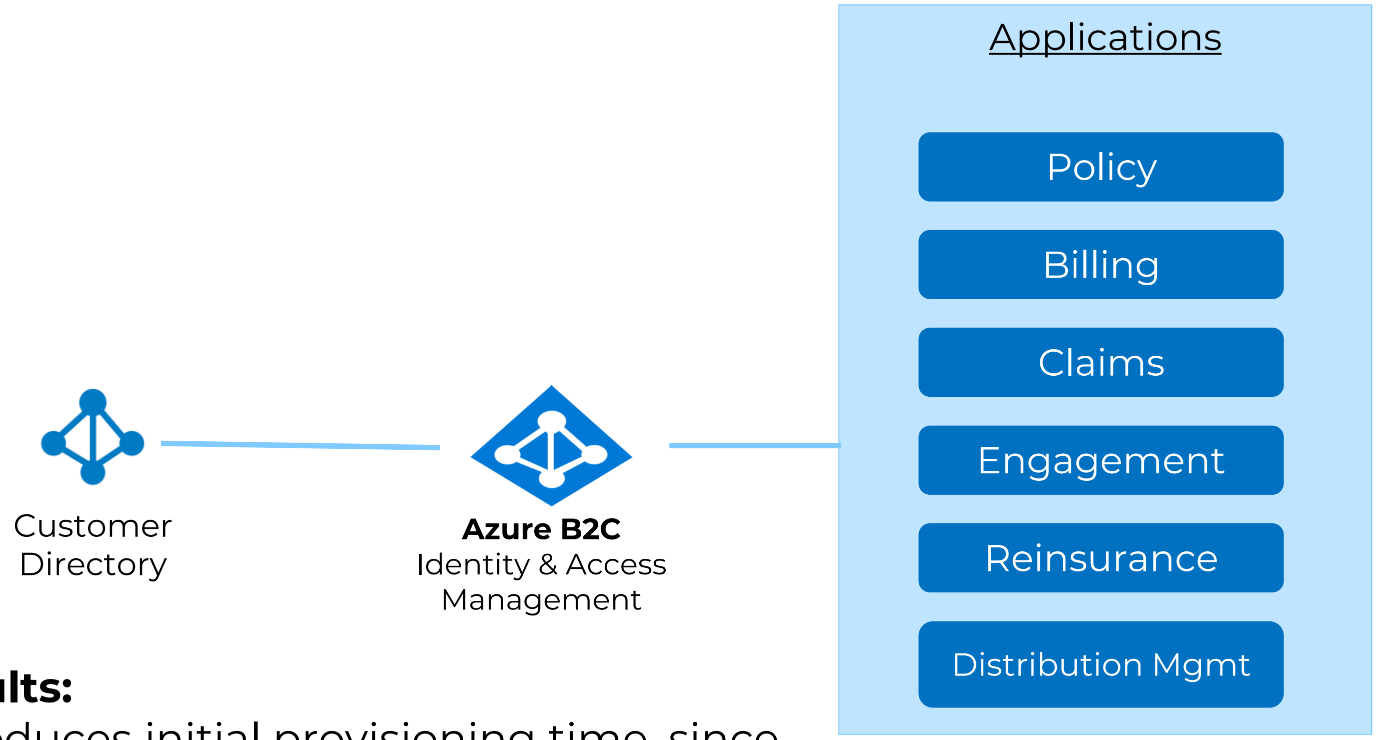
Why:
Clean separation between DC code and customer code

Change:
In the past, this was done by deploying custom binaries together with the duck creek applications. This often introduced build dependencies, complicating upgrades



Results:
a) Eliminates build dependencies between Duck Creek code and customer code, easing upgrades

Azure B2C



What:

Azure B2C a cloud-based identity and access management solution

Use:

Used to mediate identity and access management between DC systems and customer identity providers

Why:

Easily federate identity

Change:

In the past, this was done via identity plug-ins provisioned and configured with each customer-specific instance of the Duck Creek application. Now the DC applications have a single, uniform connection to Azure B2C, where client-specific configurations are managed.

Results:

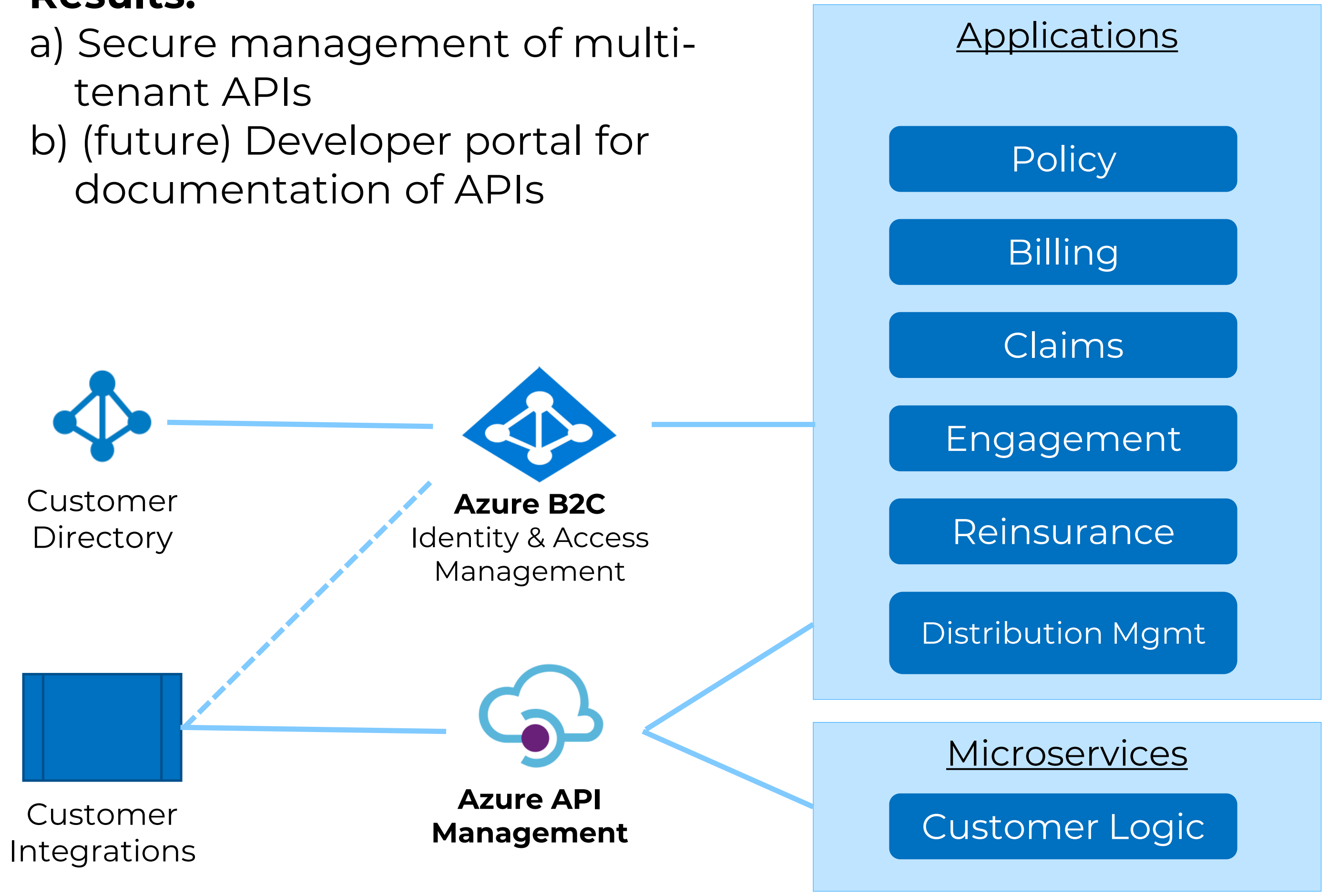
a) Reduces initial provisioning time, since this is now pure configuration
b) Removed tenant-specific plug-ins and configuration from the DC applications, making them tenant-agnostic.

Azure API Management

Results:

a) Secure management of multi-tenant APIs

b) (future) Developer portal for documentation of APIs



What:

Azure API Management is a cloud service used to publish, secure, transform, maintain, and monitor APIs.

Use:

Manage and secure APIs

Why:

Allows secure use of APIs in multi-tenant environment

Change:

In the past, Anywhere APIs were published per-tenant.

Duck Creek Application Hosting

What:

Azure AKS is a managed container orchestration service, based on Kubernetes

Use:

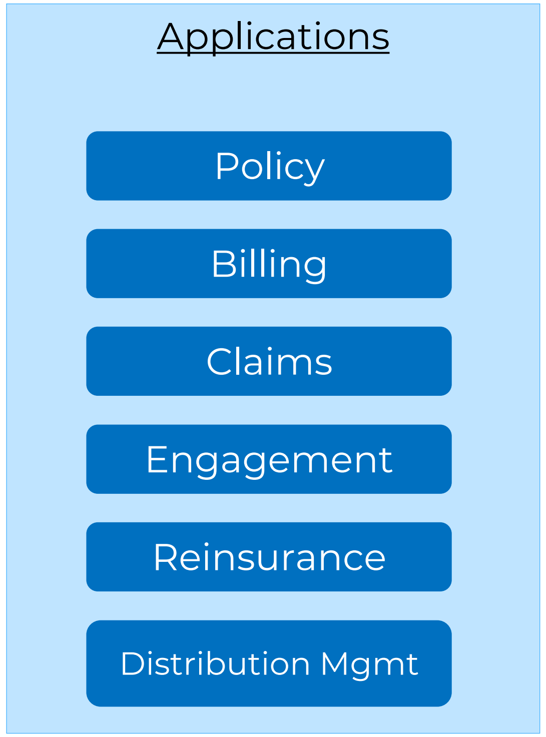
Used to host containerized applications

Why:

Highly efficient. Fast start-up allows auto-scaling.

Change:

In the past, applications were hosted on tenant-dedicated VMs. Long start-up times prevented auto-scaling. This in turn led to inefficient use of resources due to over-provisioning.

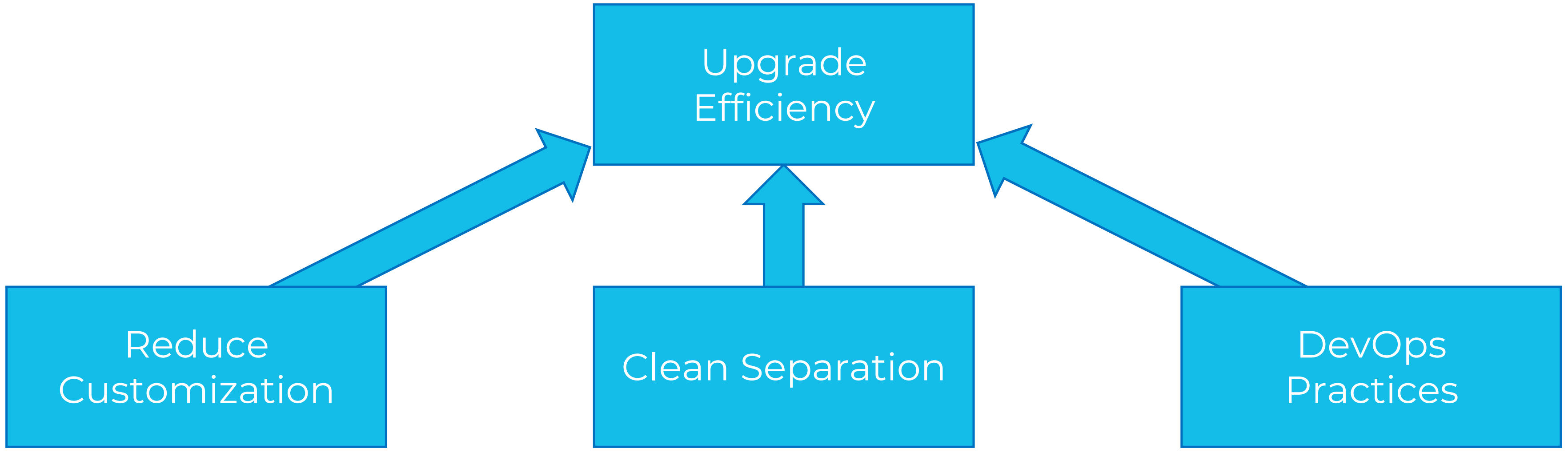


Results:

a) Reduces startup time, enabling auto-scaling

b) Improved scalability

Implementation Guidelines



Where possible, replace customization with configuration

Do not mix client code/configuration with Duck code/configuration

Automated Testing,
Automated Deployment
Telemetry
Change Management