Infrastructure Forecast: Cloudy with a Chance of APIs

Duncan Lawie
Head of DBaaS Architecture and DevOps
Group CTO Strategy, Architecture & Transformation

June 26, 2019

CREDIT SUISSE

CREDIT SUISSE

Tensions within and across the organization
High Pressure Leading to Headwinds

- “I need to test my code fix, please deploy the code” – Application Developer
- “I can’t deploy your code, I’m busy with a Prod issue” – Infrastructure Level 1 Support

Tensions within and across the organization
High Pressure Leading to Headwinds

- “I don’t want to talk to Infrastructure … except socially” – Senior Application Development Lead
- “I can’t fix my infrastructure problem – I need a Sys Admin to help” – Junior Application Developer

Tensions within and across the organization
High Pressure Leading to Headwinds

- “API First” – Senior Infrastructure Architect
- “My team just delivered a GUI for infrastructure provisioning” – Infrastructure Team Lead

Tensions within and across the organization
High Pressure Leading to Headwinds

- “Prod is broken, just let me log in and fix it” – Infrastructure Level 2 Support
- “This is a SOX-relevant application, you can’t just log in” – Security Operations

Tensions within and across the organization
High Pressure Leading to Headwinds

- “My database is running slow – please tell me why” – Every Application Support Team
- “It’s not the database, it’s your application” – Every DBA

My Journey
There's no bad weather, just the wrong clothing

- Once, there was only Prod
- The Cult of the DBA
- Developer Productivity
- Collaboration

Starting Point: Who are Ops?
Traditional Answer: Anyone who can access Prod

- Who can access Production?
  - Infrastructure Staff
  - Application Developers
  - Application Support

Starting Point: Who are Ops?
Traditional Answer: Anyone who can access Prod

- Are Infrastructure teams really Ops?

Target: Infrastructure as Code
or “no standing access to production” or “NoOps”

- Infrastructure staff manage the environment, not the applications
- Application Developers develop and test the application, using common tooling
- Application Support deploy and manage the application, using the same tooling

Target: Infrastructure as Code
or “no standing access to production” or “NoOps”

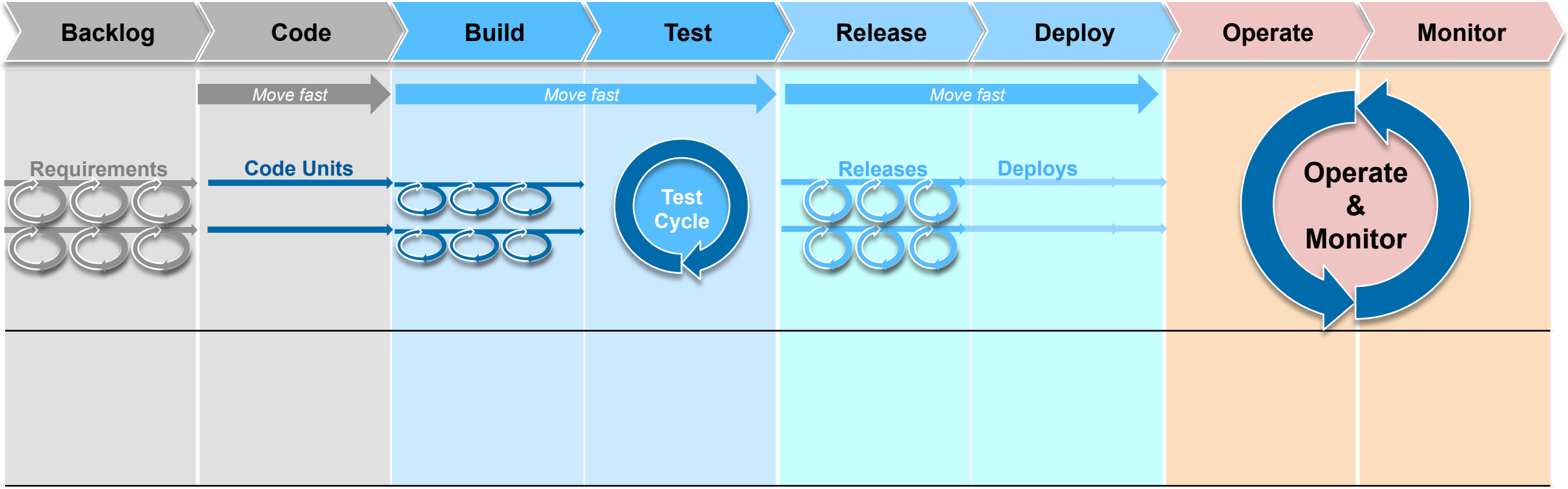
- Infrastructure teams build tools and platforms that Application DevOps use to interact with Infrastructure …

Cloud!

Where we have got to, So far
Sea State: Moderate; Visibility: Good

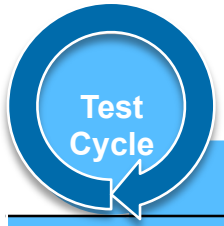
- Progress on concepts and culture
- Progress on automation
- Progress on APIs

Progress: Understanding the Software Delivery Pipeline
and what Infrastructure supplies

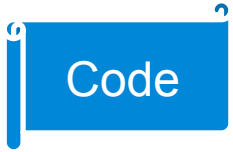


How Can Infrastructure Support the Development Pipeline?

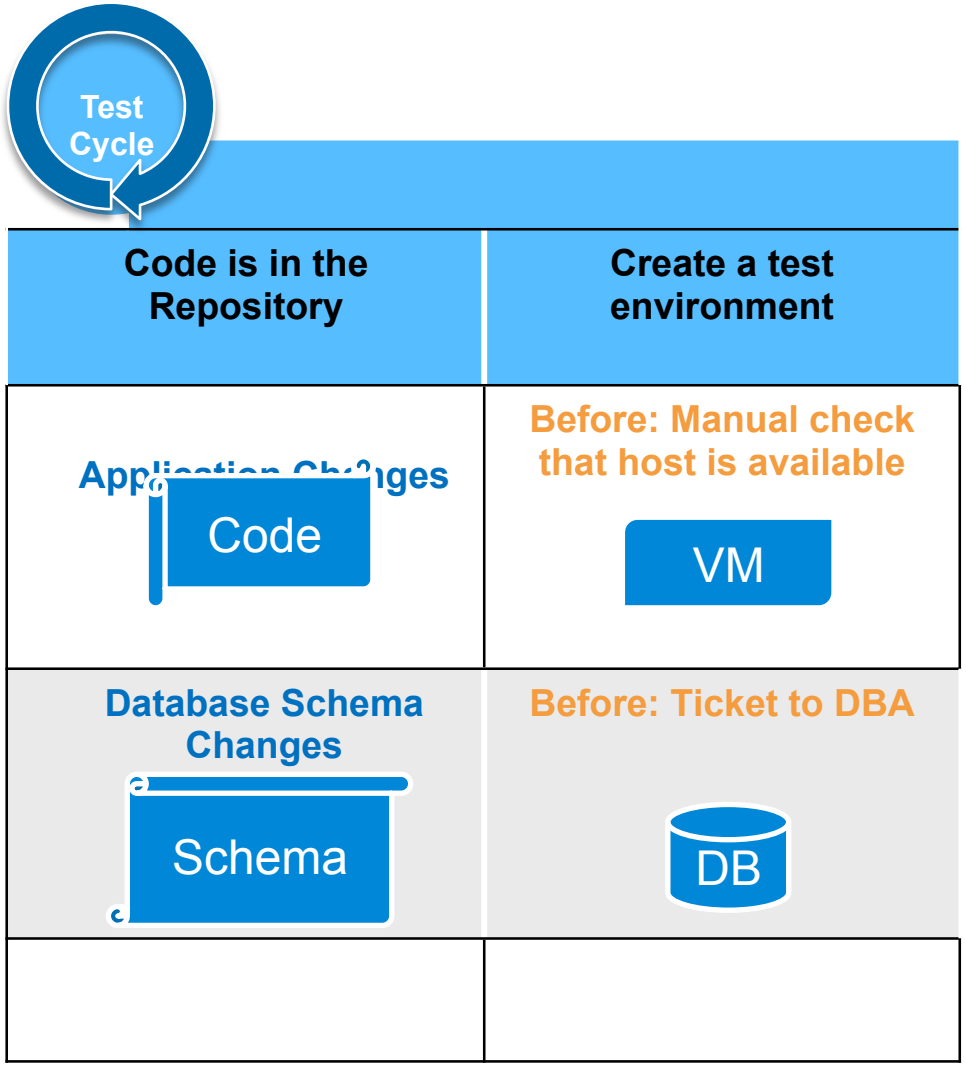
Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs



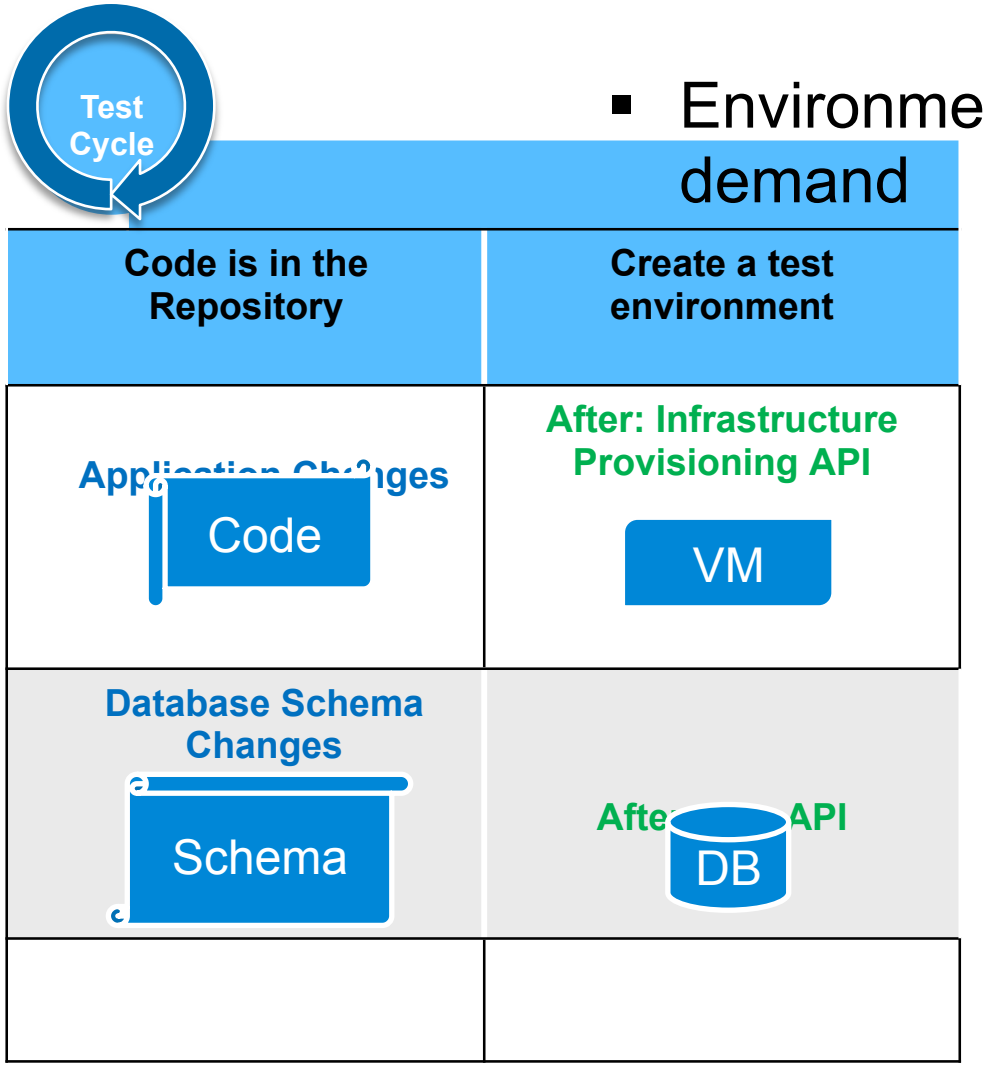
Workflow

Code is in the Repository	Create a test environment	Deploy the code	Execute the tests	Clear down the test environment
Application Changes 				
Database Schema Changes 				

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs



Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs



- Environment provided on demand

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs

Code is in the Repository	Create a test environment	Deploy the code
Application Changes Code	VM	Before: Application specific tooling
Database Schema Changes Schema	DB	Before: Ticket to DBA

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs

- Process remains within one toolset

Code is in the Repository	Create a test environment	Deploy the code
Application Changes 		After: Tooling common across all Java applications
Database Schema Changes 		After: Liquibase Integration built into common tooling

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs

Code is in the Repository	Create a test environment	Deploy the code	Execute the tests
Application Changes 			Before: Application specific (or manual)
Database Schema Changes 			Before: Often mocked, for early test cycles

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs

- More complete testing in each cycle

Code is in the Repository	Create a test environment	Deploy the code	Execute the tests
Application Changes 			After: Orchestrated through the pipeline
Database Schema Changes 			After: Can be fully tested in all test cycles

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs

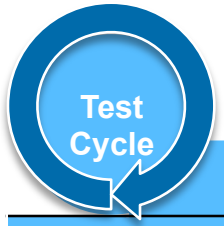
Code is in the Repository	Create a test environment	Deploy the code	Execute the tests	Clear down the test environment
Application Changes 				Before: Scripted cleanup of a host
Database Schema Changes 				Before: Left for next time, scripted cleanup or ticket to DBA

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs

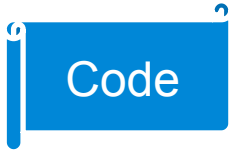
- Resources released for re-use elsewhere

Code is in the Repository	Create a test environment	Deploy the code	Execute the tests	Clear down the test environment
Application Changes 				After: Infrastructure API to delete host
Database Schema Changes 				After: Use API to clear DB

Progress: Impediments Reduction in the Test Cycle
Benefit of Private Cloud, CI/CD Integrations, APIs



Workflow

Code is in the Repository	Create a test environment	Deploy the code	Execute the tests	Clear down the test environment
Application Changes 	Before: Manual check that host is available After: Infrastructure Provisioning API 	Before: Application specific tooling After: Tooling common across all Java applications	Before: Application specific (or manual) After: Orchestrated through the pipeline	Before: Scripted cleanup of a host After: Infrastructure API to delete host
Database Schema Changes 	Before: Ticket to DBA After: Use API 	Before: Ticket to DBA After: Liquibase Integration built into common tooling	Before: Often mocked, for early test cycles After: Can be fully tested in all test cycles	Before: Left for next time, scripted cleanup or ticket to DBA After: Use API to clear DB
Benefits: Faster, lower cost, more complete	Environment provided on demand	Process remains within one toolset	More complete testing in each cycle	Resources released for re-use elsewhere

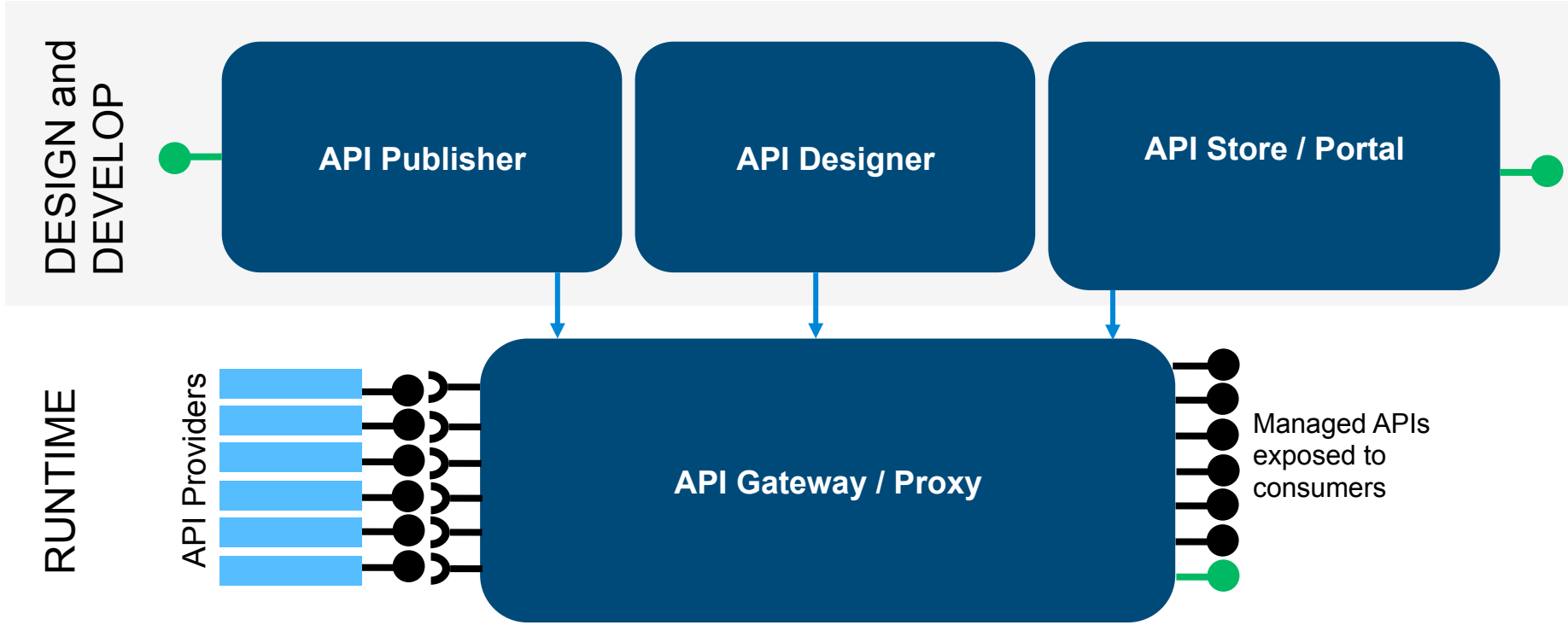
Progress: Infrastructure API Marketplace

A new delivery model for services – API First

KEY
API management functionality exposed as APIs

Progress: Infrastructure API Marketplace

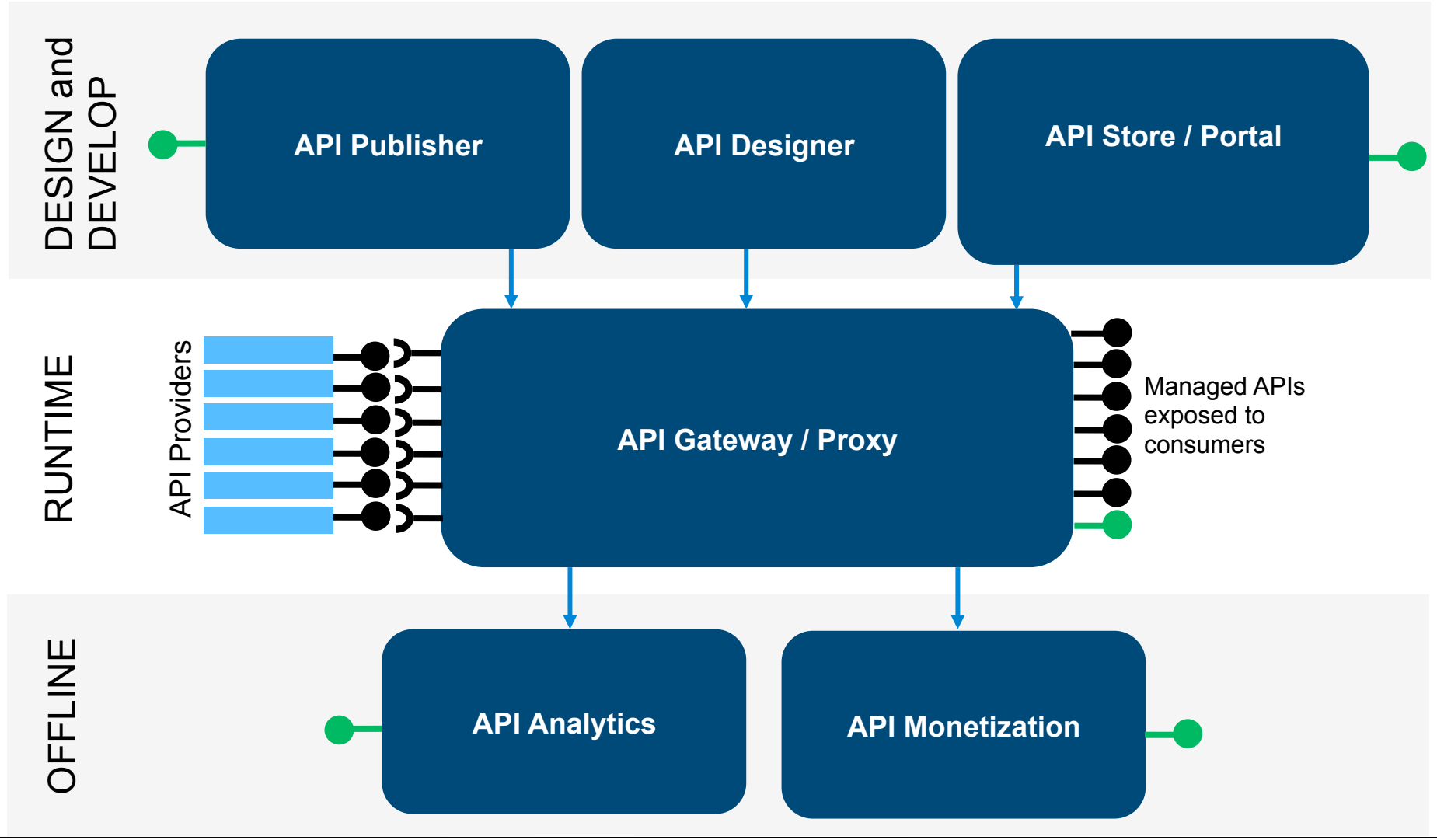
A new delivery model for services – API First



KEY
API management functionality exposed as APIs

Progress: Infrastructure API Marketplace

A new delivery model for services – API First



Progress: Using a Vendor API improves responsiveness
and reduces maintenance effort for us

- Using Vendor API
- Aligning Credit Suisse and Vendor strategy
- 98% Faster for API users

Progress: Automation of Infrastructure Patching
Leading to a more resilient environment

LOW FREQUENCY

Patching conducted quarterly or on a need basis

HIGHER FREQUENCY

Increased frequency of patching with same resources

LOW COMPLIANCE

Lower compliance due to infrequent patching

HIGH COMPLIANCE

Better compliance achieved by regular patching

Progress: Automation of Infrastructure Patching
Leading to a more resilient environment

LOW ACCURACY

Frequent
human errors

HIGH ACCURACY

Improved
Security Patch
Compliance

HIGH MANUAL EFFORT

Additional time
spent manually of
high cost
workforce

LOW MANUAL EFFORT

Less time spent
on weekends by
IT support teams

Progress: Automation of Infrastructure Patching
Leading to a more resilient environment

NO AUDIT TRAILS

No consistent Audit Trail for patching

AUDIT TRAILS

Maintenance of an Audit Trail of all actions performed

MISSED SLAs

Not achieving the SLA of patching compliance

ACHIEVED SLAs

Ability to adhere to highest SLA Compliance

Progress: Automation of Infrastructure Patching
Leading to a more resilient environment

BEFORE

LOW FREQUENCY

Patching conducted quarterly or on a need basis

LOW COMPLIANCE

Lower compliance due to infrequent patching

LOW ACCURACY

Frequent human errors

HIGH MANUAL EFFORT

Additional time spent manually of high cost workforce

NO AUDIT TRAILS

No consistent Audit Trail for patching

MISSED SLAs

Not achieving the SLA of patching compliance

AFTER

HIGHER FREQUENCY

Increased frequency of patching with same resources

HIGH COMPLIANCE

Better compliance achieved by regular patching

HIGH ACCURACY

Improved Security Patch Compliance

LOW MANUAL EFFORT

Less time spent on weekends by IT support teams

AUDIT TRAILS

Maintenance of an Audit Trail of all actions performed

ACHIEVED SLAs

Ability to adhere to highest SLA Compliance

Outcomes: There is Value from this Work
It’s going to be a bright, sunshiny day

- Empathy
- Improved Cycle Time for Development
- Continued Reliability, Resilience and Availability of Infrastructure
- Security Maintained. Or Improved

Infrastructure Forecast:
APIs, Self-Service and Cloud-like Platforms