Cloud Native Application Development

A new focus on compute to the cloud

AGENDA

1. Speaker Intro
2. The Strategic Shift
3. Defining Cloud Native
4. Understanding Cloud Native Development
5. Operational Excellence
6. The Roadmap Ahead (Demo)
7. Q&A

SPEAKER INTRO

Chief Software Architect

Enterprise Architecture Group |
Information Technology Services

- 35 years of Software Engineering Experience
- Employment at 4 UC Campuses
- Various Cloud Platform Certified

UC SANTA BARBARA

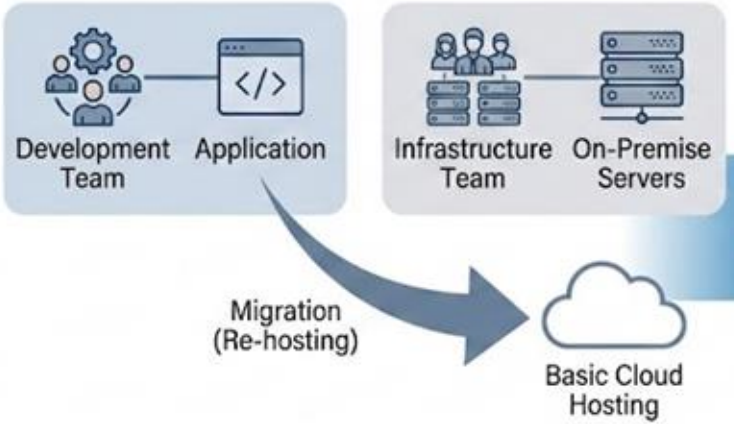
The Strategic Shift

UCSB ITS

UC SANTA BARBARA

Our Shared Journey: Evolving Beyond On-Premise and Lift & Shift

Historical Approach: On-Premise & "Lift and Shift"



- Acknowledging our history of on-premise development and "lift and shift" strategy.
- Limited application scalability and infrastructure efficiency.
- Traditional on-premise silos persist.

Strategic Shift Towards Cloud-Native Practices

The Imperative: Collaborative, Cloud-Native Future

- Infrastructure considerations become integral to the application development lifecycle.
- Builds modern, scalable, and resilient cloud applications.
- Embraces truly cloud-native practices.

Where We Have Been:

Where We Have Been:

- Our history is rooted in on-premise application development and “lift and shift” migration strategies.
- The Limitation: While ‘lift and shift’ moved us to the cloud, it failed to unlock true application scalability and infrastructure efficiency.

Where We Are Going:

Where We Are Going:

- We are moving away from traditional silos toward a collaborative, cloud-native future.
- Infrastructure considerations must now become integral to the application development lifecycle.

What is Cloud Native

CORE CHARACTERISTICS OF CLOUD NATIVE

MODULAR ARCHITECTURE

Service-based design rather than monolithic structures.

STANDARDIZED CONTAINERIZATION

Consistent packaging across environments.

AUTOMATED MANAGEMENT

Utilizing platforms for automated resource handling.

INTEGRATED DEVOPS

Merging development and operations practices.

KEY TAKEAWAY: These are not just application concerns; they have profound implications for the underlying infrastructure.

PILLAR 1 – MODULAR SERVICE-BASED ARCHITECTURE

CONCEPT

Breaking down monolithic applications into independently deployable services.

INFRASTRUCTURE ALIGNMENT

The infrastructure must support a distributed nature. Requires changes to network configurations, load balancing, and inter-service communication. This represents a significant departure from traditional on-premise network architectures.

PILLAR 2 – STANDARDIZED CONTAINERIZATION

THE FOUNDATION

Using standardized units for consistent packaging and deployment across development and operations.

INFRASTRUCTURE AS CODE (IaC)

Containerization facilitates IaC principles. Allows infrastructure teams to manage resources programmatically to support specific application needs.

PILLAR 3 – AUTOMATED RESOURCE MANAGEMENT

COLLABORATIVE SCALING

Utilization of platforms to automate deployment, scaling, and management.

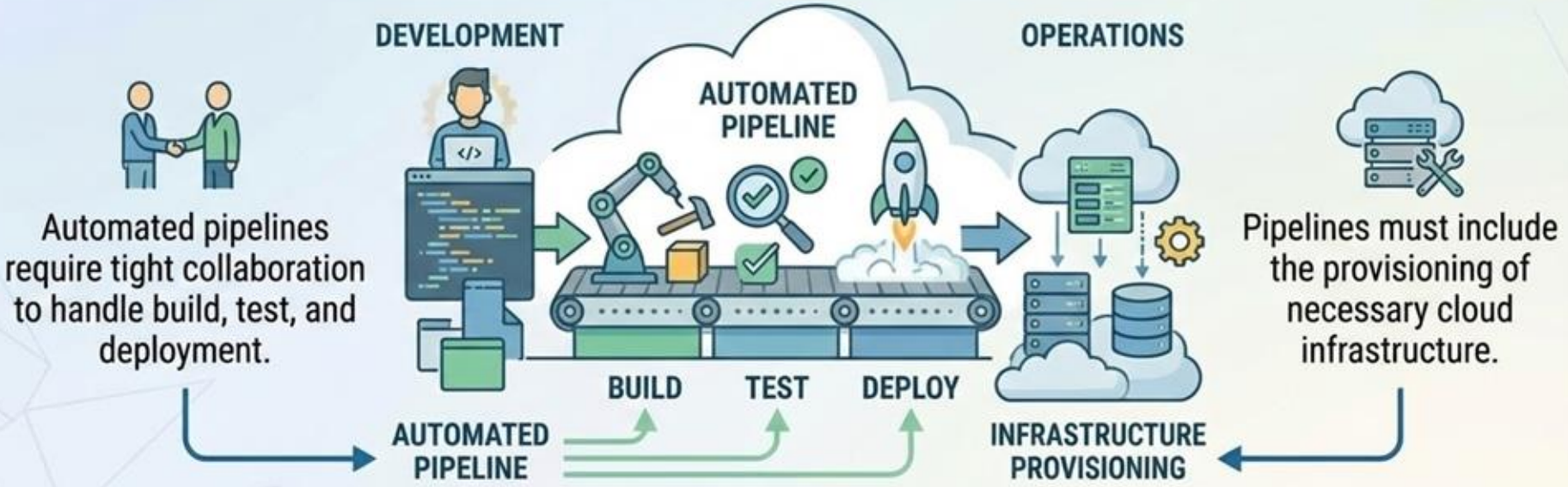
TEAM SYNERGY

Developers and Infrastructure Engineers must collaborate on resource definitions.

Joint understanding of scaling policies and network configurations is essential.

PILLAR 4 – INTEGRATED DEVOPS

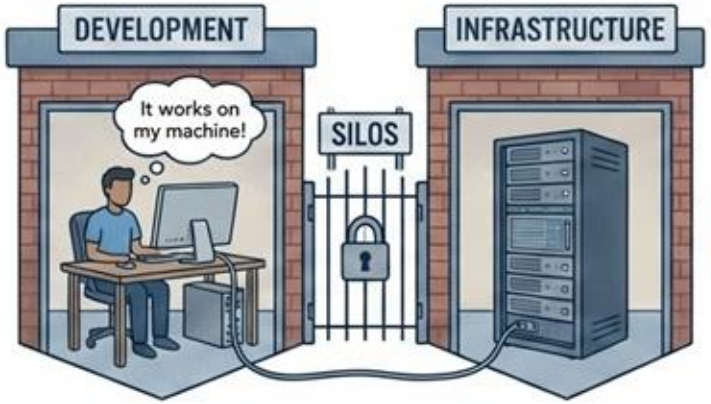
Bridging the Gap: Automation is the key to the cloud-native lifecycle.



Understanding Cloud Native Development

ON-PREMISE (Traditional):
Silos & Static Environments

CLOUD NATIVE (Modern):
Infrastructure Awareness & Dynamic Resources



THE FUNDAMENTAL SHIFT

- Static Environments
- Local Debugging (LAN-restricted)
- Unaware of Server Constraints

INFRASTRUCTURE AS CODE (IaC):
Developer Provisioning, Reduced Friction

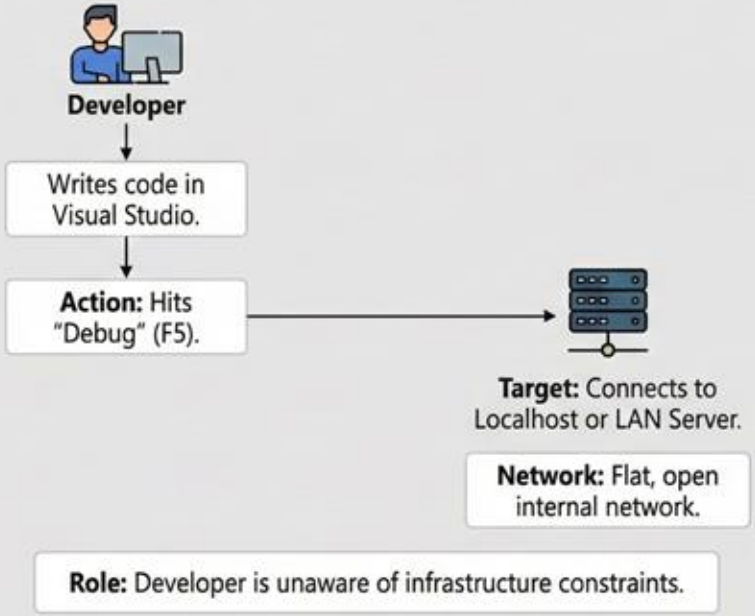
- Infrastructure Awareness
- Remote Debugging with Cloud Tools
- Distributed Network Boundary
- Identical Dev/Test Environments

Development Workflow Evolution

From Local Silos to Infrastructure Awareness

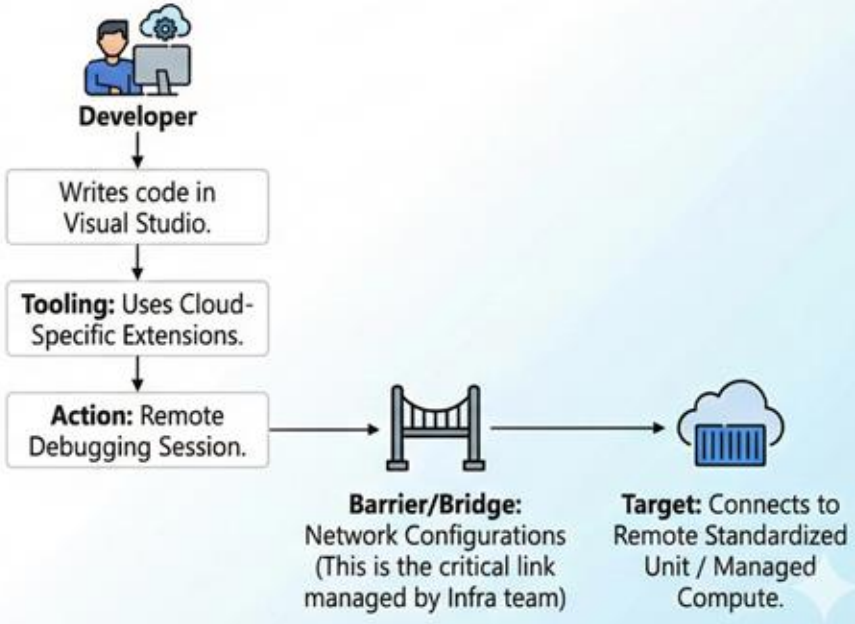
The Traditional "Silo" (On-Premise)

Local & Static



The Cloud Native "Collaboration"

Remote & Infrastructure Aware



1. THE FUNDAMENTAL SHIFT

On-Premise
(Traditional)

Siloed development, static environments. Unaware of physical constraints.

Cloud Native
(Modern)

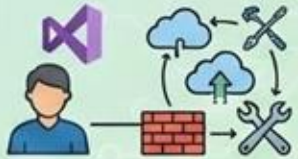
Infrastructure Awareness. Code interacts with dynamic resources.

2. EXAMPLE: DEBUGGING

The Old Way
(On-Premise)

Local debugging, simple network access.

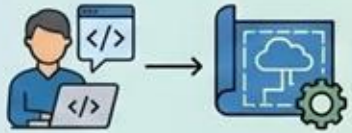
The New Way
(Cloud Native)



Remote debugging, Cloud Extensions, Infrastructure Dependency.

3. INFRASTRUCTURE AS CODE (IaC)

Developer Provisioning



Provision infrastructure with declarative config.

Reduced Friction

Identical environments, no “it works on my machine” issues.

UC SANTA BARBARA

OPERATIONAL EXCELLENCE

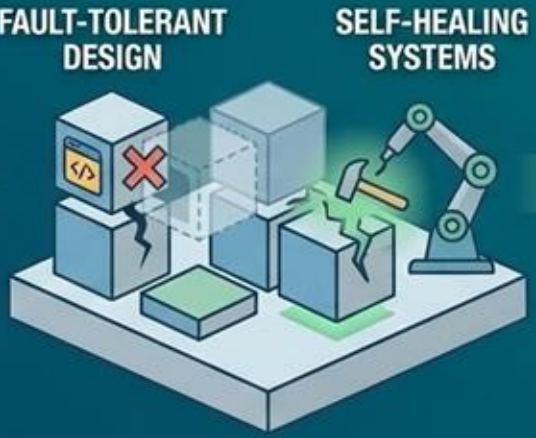
UC SANTA BARBARA

BUILDING RESILIENT APPLICATIONS

SHARED RESPONSIBILITY

Resilience is not solely an application concern.

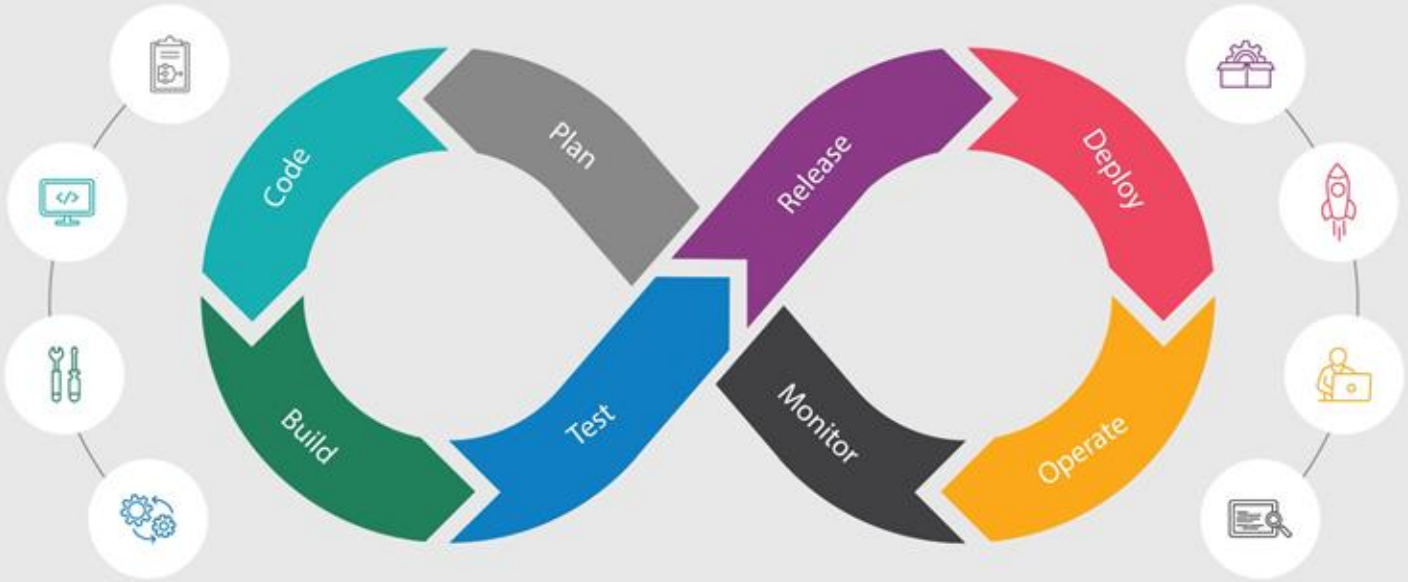
STRATEGIES



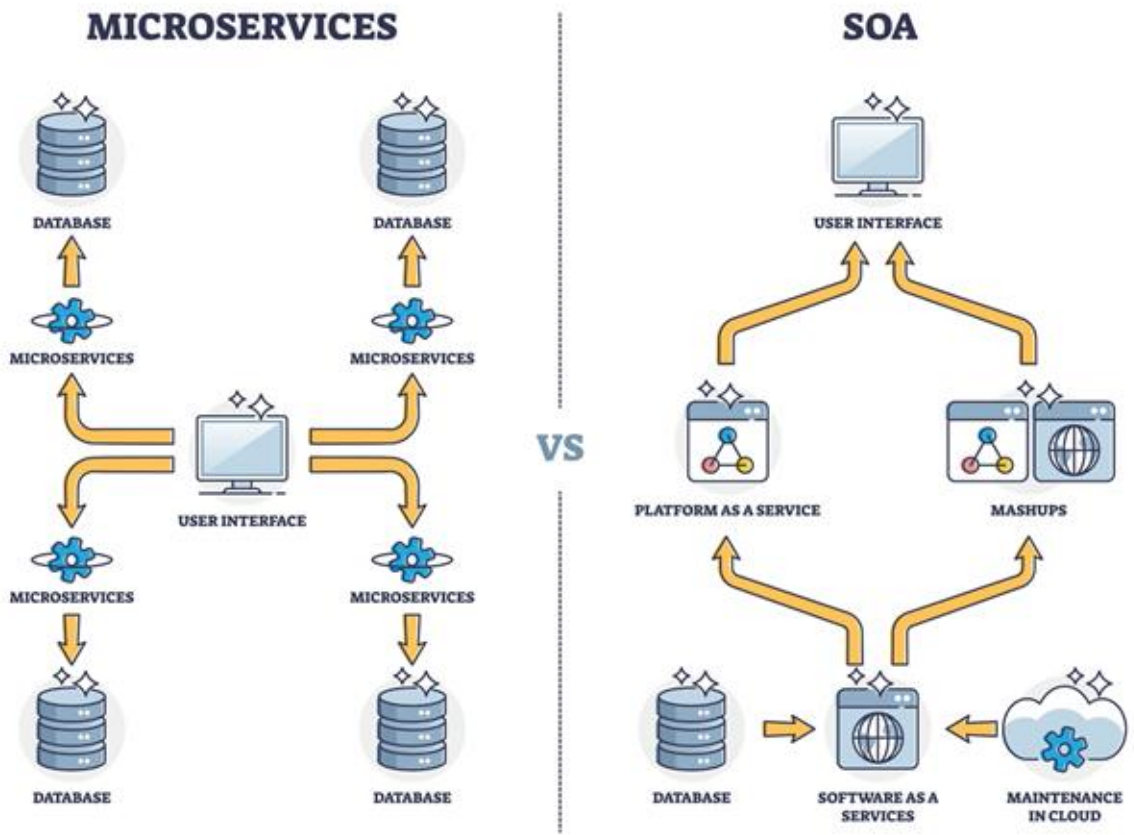
Designing fault-tolerant and self-healing systems.

IMPLEMENTATION

Requires correct configuration of infrastructure components, such as auto-scaling and availability zones.



DevOps Process


MICROSERVICES
DATABASE
DATABASE
MICROSERVICES
MICROSERVICES
USER INTERFACE
MICROSERVICES
MICROSERVICES
DATABASE
DATABASE
VS
SOA
USER INTERFACE
PLATFORM AS A SERVICE
MASHUPS
DATABASE
SOFTWARE AS A SERVICES
MAINTENANCE IN CLOUD

The Roadmap Ahead

Strategic Goal: Cloud-Native Transition

The Strategic Goal

Implementing practical steps and considerations for our organization to strategically transition towards real-world cloud-native development practices.

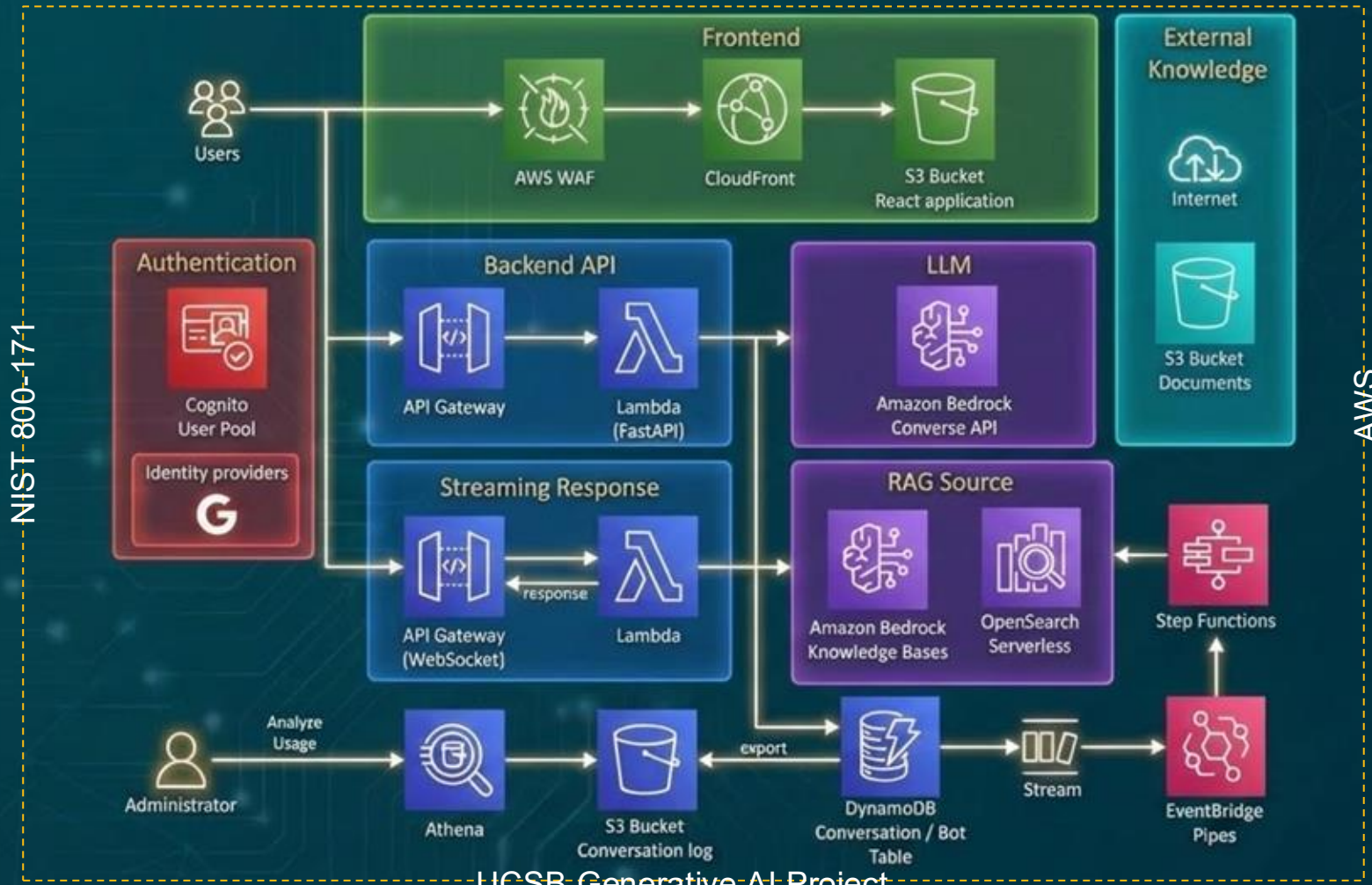
The Critical Foundation

Success emphasizes the absolute need for shared understanding across technical disciplines.

Bridging the Teams

The roadmap relies on deep collaboration between Application Development and Infrastructure Engineering teams, moving away from isolated silos.

UCSB AI Initiative : A Cloud Native Application



UCSB Generative AI Project

Questions/Discussions

Excellence | People-Focused | Integrity | Innovation | Collaboration

UC SANTA BARBARA