CONFLUENT

Multi-cloud & the Chamber of Secrets

Michael Kehoe
Sr Staff Security Engineer

Agenda

01 $ whoami

02 Background of Confluent’s infra

03 Problem Introduction

04 Defining a secret strategy

05 Implementing a secret policy

06 Implementing controls

07 Conclusion

08 Q & A

$ whoami

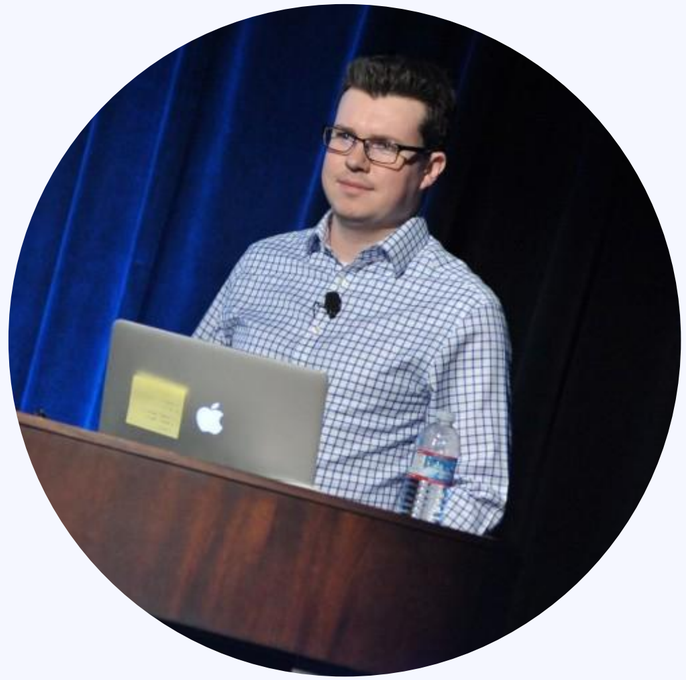
$ whoami

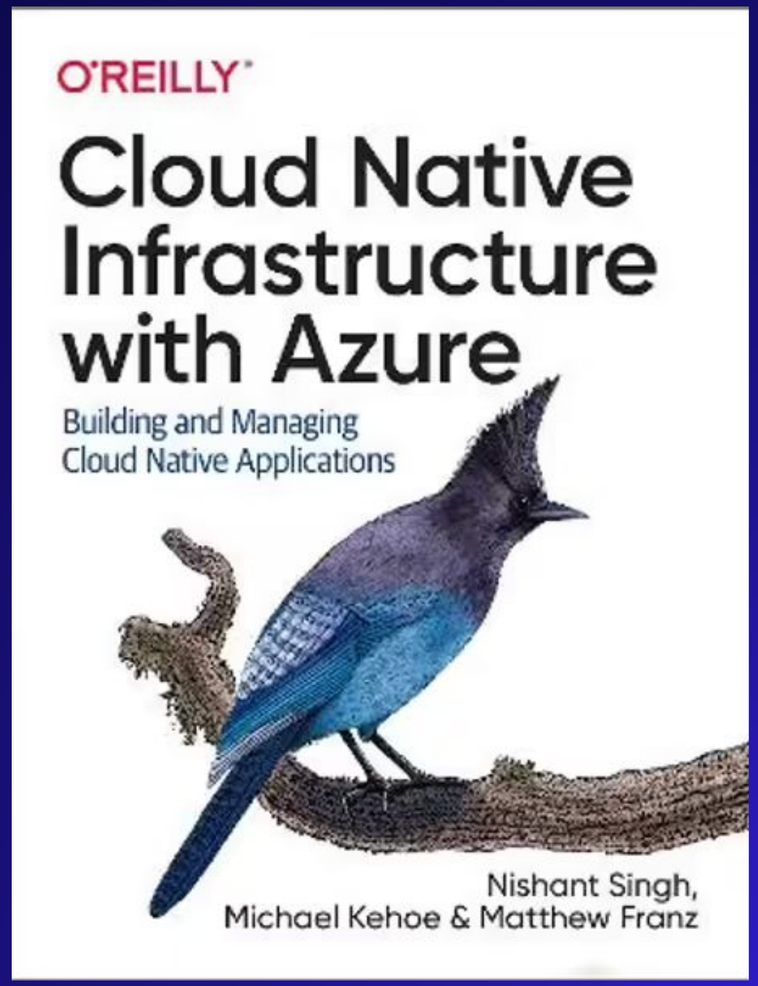
- Sr Staff Security Engineer - Confluent
    - Cloud Architecture & Reliability

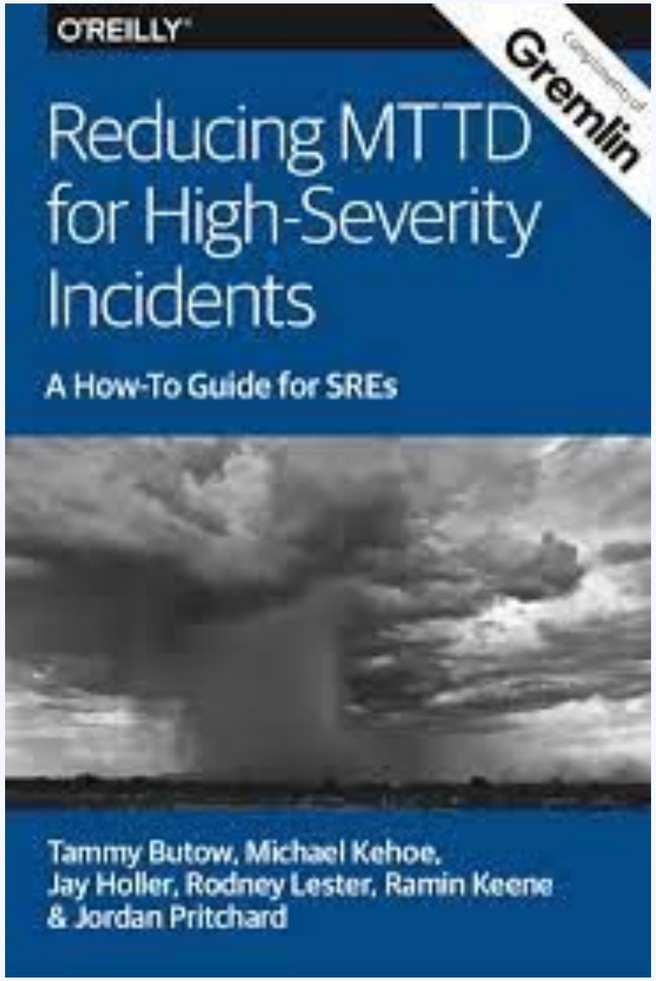
- Previously:
    - Sr Staff SRE @ LinkedIn
    - PhoneSat intern @ NASA

- Background in:
    - Networks
    - Microservices
    - Traffic Engineering
    - KV Databases
    - Incident Management

- Twitter: @michaelkkehoe
- LinkedIn: linkedin.com/in/michaelkkkehoe
- Website: michael-kehoe.io

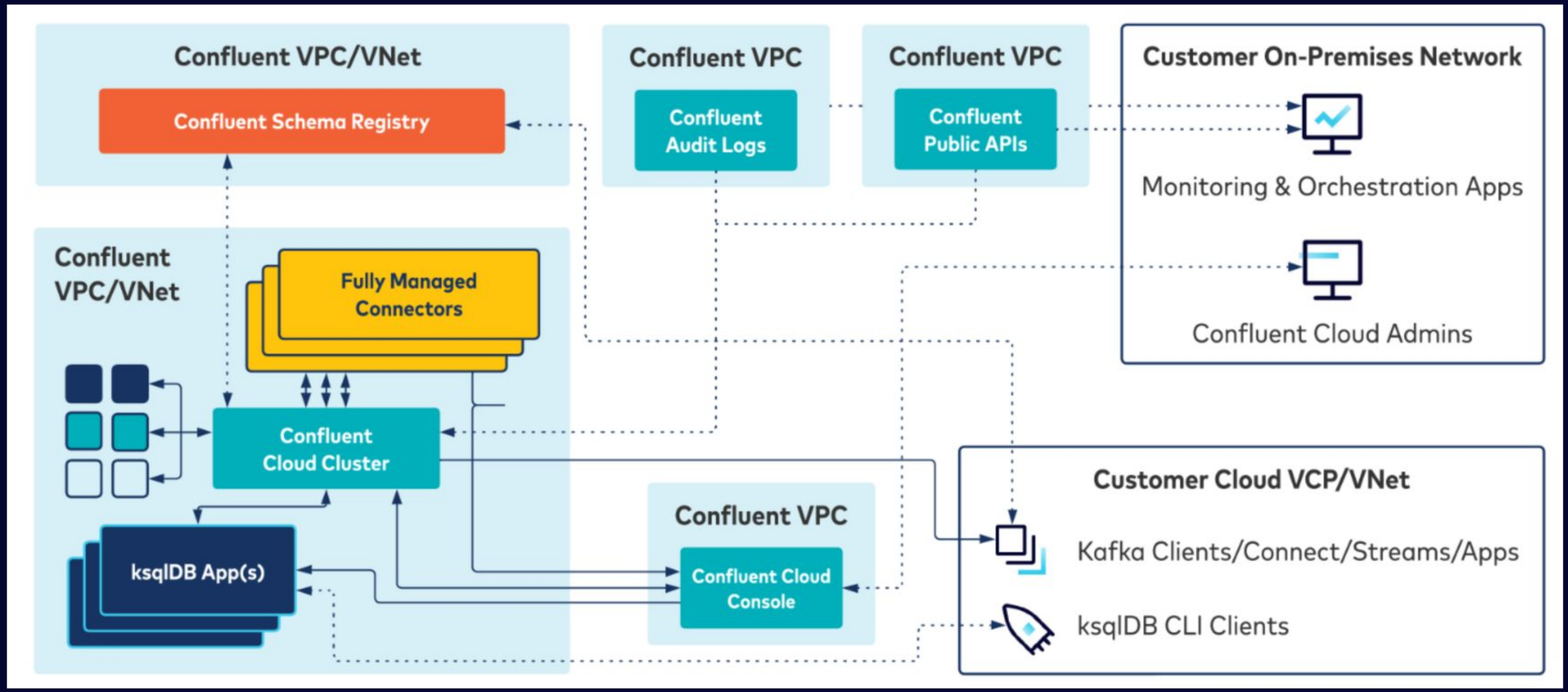



O'REILLY
Cloud Native Infrastructure with Azure
Building and Managing Cloud Native Applications
Nishant Singh, Michael Kehoe & Matthew Franz

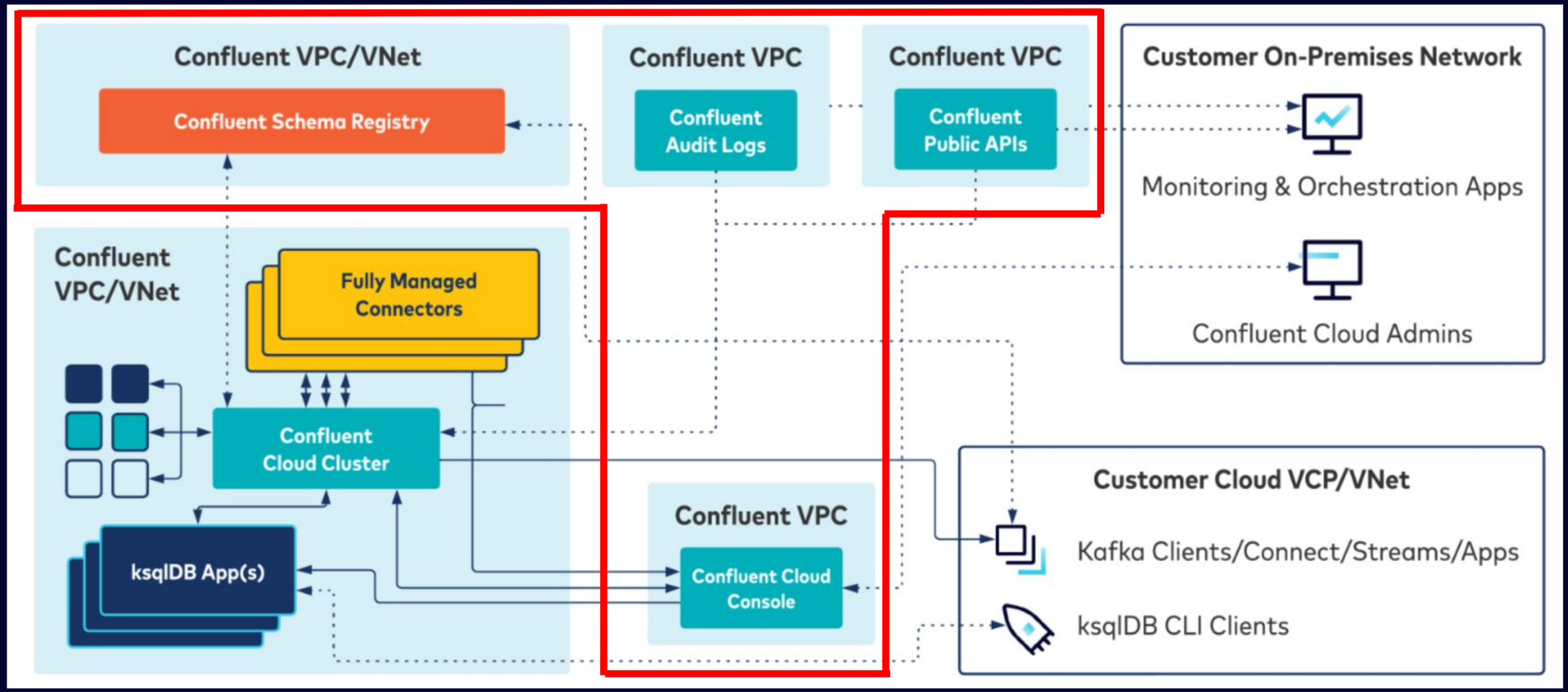

O'REILLY
Compliments of Gremlin
Reducing MTTD for High-Severity Incidents
A How-To Guide for SREs
Tammy Butow, Michael Kehoe, Jay Holler, Rodney Lester, Ramin Keene & Jordan Pritchard

Confluent's Architecture

Confluent Architecture

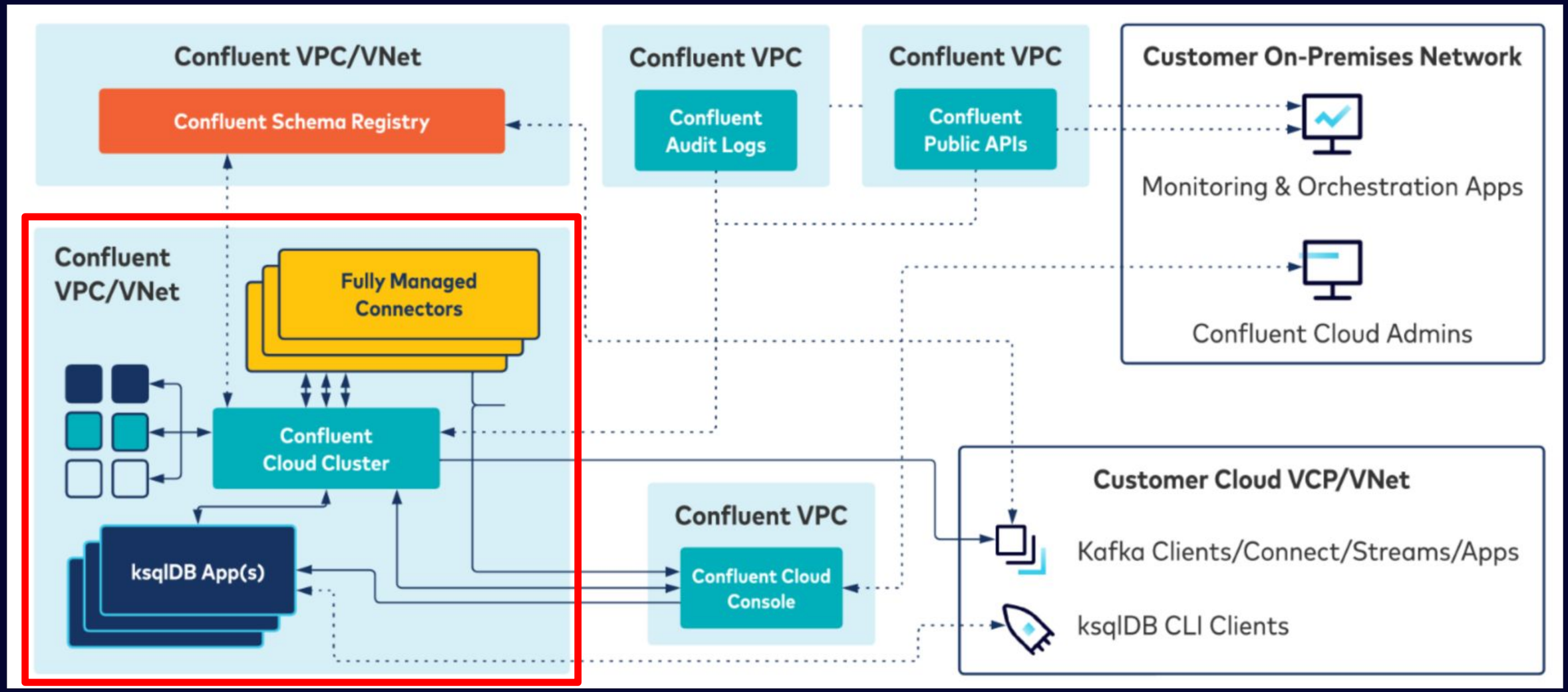


Confluent Architecture - Control Plane



AWS

Confluent Architecture - Customer infra



x thousands across 3 CSPs

Problem Statement

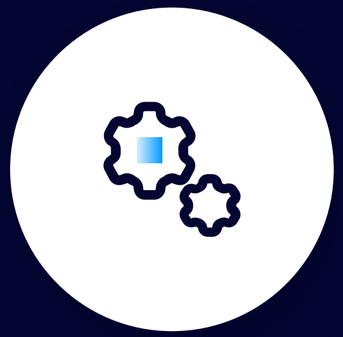
Problem Statement

Global Multi-Cloud

AWS, Azure, GCP over multiple regions

3rd party secrets

Some of the secret mechanisms we control, in some cases, we have secrets for 3rd party services



Control-plane vs data-plane

We need to be able to serve secrets in our control-plane & customer data-plane infra

Ownership

Finding who owns a secret is hard

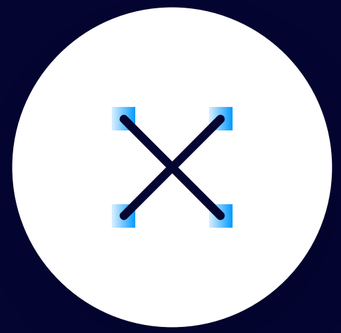
Defining a secret strategy - Finding the blind spots

Finding the blind spots

01	AWS	• AWS Secrets Manager • AWS KMS • AWS IAM Roles/ Users
02	GCP	• GCP IAM Service Accounts • GCP secrets manager
03	Azure	• Azure KeyVault • Azure Service Principals • Azure Managed Identities
04	Hashicorp Vault	• Database credentials • Internal API keys • 3rd party API keys
05	3rd party systems	• API Keys • Critical IT/ Business systems

Defining a secret strategy - Building an inventory

Building an inventory - Take 1



Manually retrieve inventory

Manually pull an inventory from each system and collate it into a spreadsheet

Parse permissions/ policy

Manually retrieve policy/ permission and evaluate security/ value of secrets

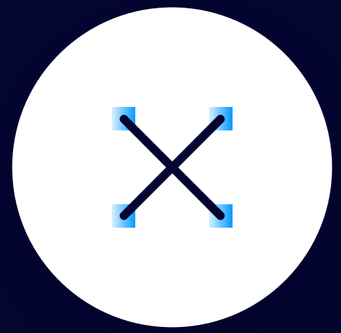
Find owners

Find who owns the secret

File tickets for remediation

Ask owners to update information, add controls

Building an inventory - Take 2



Automated inventory retrieval

Use DivvyCloud/ Steampipe to create inventory

Parse permissions/ policy

Automated parsing of policy and permissions against predetermined risk levels

Find owners

All secrets now have metadata for ownership

File tickets for remediation

Based on metadata, automated tickets can be filed

Implementing a secret policy - Defining a strategy

Creating a Strategy

Define what is high-value

Create a definition of what we consider keys-to-the-kingdom and inventory against the sensitivity of the credential

Implement policy for secret lifetimes

For security and compliance reasons, implement a static credential rotation policy

Define approved systems/ controls

Define what credential types should be used and what controls they require

Force ownership

Ensure that every secret has defined ownership.

- Use of IaC in Vault
- Use of IaC (tags) with CSPs

Utilize the best of Vault

1. Utilize dynamic engines as much as possible
2. Terraform IaC against Vault

Improved inventory & monitoring

Daily inventory of all known secrets & usage monitoring of select HV secrets.

Implementing a secret policy - Allowing exceptions

Allowing exceptions

Because only a sith deals in absolutes...

There will always be exceptions

There will always be an edge-case that needs to be accounted for

Create alternative controls

If a secret can not have a preventative control... create a monitoring control

Inventory your exceptions

Ensure that you document the exception

Implementing secret controls - Preventative Controls

Building preventative controls

For the best case

Utilize Vault engines

Constantly rotate secrets using dynamic engines (we built our own engines)

Utilize CSP native dynamic identities

Make the CSP responsible for managing the credential

Utilize IP restrictions

In the unideal case, place IP restrictions on the use of the credential

Implementing secret controls - Building Monitoring controls

Building monitoring controls

For when you may not be able to implement a preventive control

Utilize existing logging pipelines

Utilize the existing logging we do of our systems

Create “known usage locations”

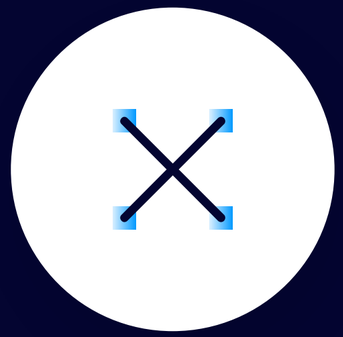
Create a list of IPs that we expect to be using the credential

Alert infosec oncall

Ensure that you document the exception

Conclusion

Conclusion



Know where your secrets are

Do a deep inventory of any place you may have a secret

Know how to secure them

Create standards for how secrets should be protected

Make secrets easy to manage

Make it easy to manage the creation/ update of the secret

Build monitoring controls

Monitor for last-use/ where they are being used from

Q & A


CONFLUENT