KubeCon CloudNativeCon North America 2026

Verkada

Hardening Kubernetes for GovCloud: A 3-Year Journey to FedRAMP Moderate Authorization

Artur Beznosyuk and Hanhong Gao, Verkada



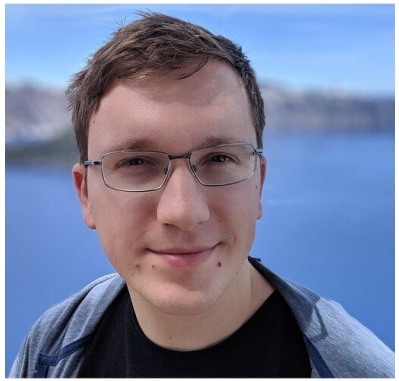
Presenters

Verkada



Hanhong Gao

VP of Engineering,
Infrastructure



Artur Beznosyuk

Software Engineer,
GovCloud Infrastructure

Agenda

1	**Why FedRAMP**	Motivation behind starting the project
2	**How it came to be**	General timeline and milestones
3	**General k8s challenges**	Nodes hardening and other requirements
4	**Vulnerabilities**	Scanning and patching vulnerabilities
5	**Deployments**	Dealing with unavailable external dependencies on the example of deployments

Disclaimer

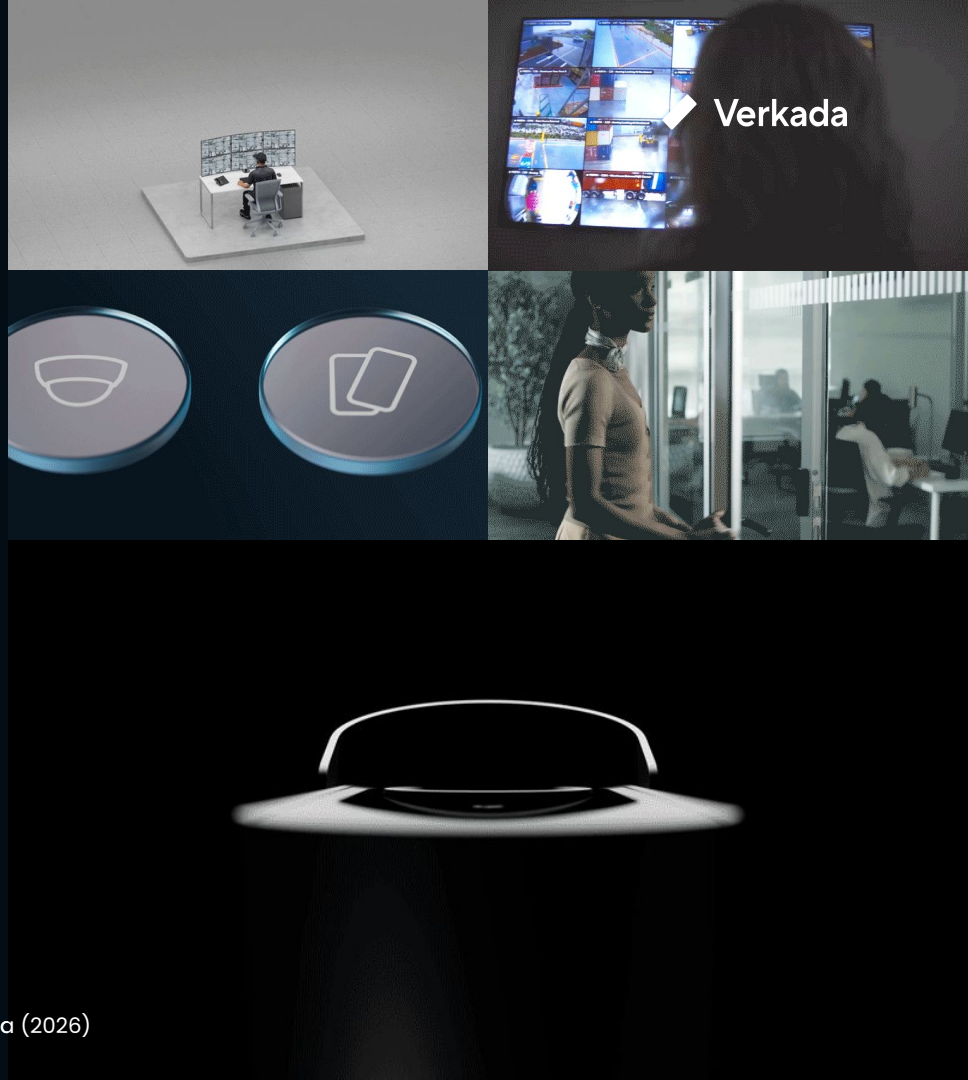
Everything presented here is a simplified and generalized version. We have extensive documentation (huge shoutout to our Security team) that covers all of the details as required by and in accordance with the FedRAMP program

Who we are

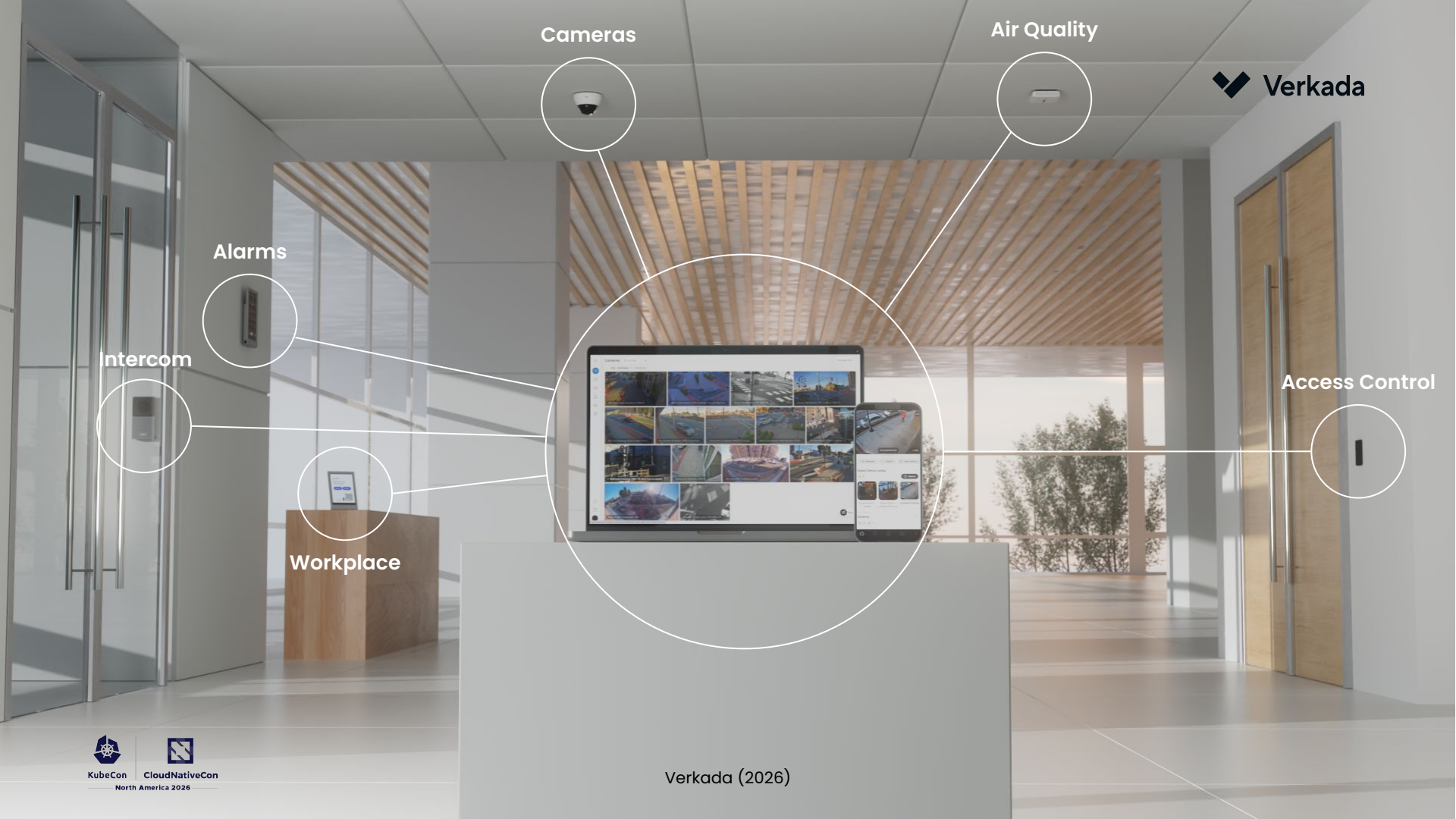
Purpose

Protect people and places in a privacy-sensitive way.

We've built an integrated platform for cloud-managed physical security.



Verkada (2026)



KubeCon CloudNativeCon North America 2026

Verkada (2026)

What is FedRAMP

- US government security baseline for cloud services, built on NIST SP 800-53 controls
- Independent auditors (3PAOs) verify it; agencies reuse the result ("do once, use many times")
- Why SaaS needs it: agencies generally can't buy cloud services without it, so it's the entry ticket to the federal market
- Not a one-time audit: it's an ongoing operational commitment

Our path: Rev5. The new path: 20x

What we went through (Rev5)

- Hundreds of NIST 800-53 Rev. 5 controls, each with a written implementation description and evidence
- Documentation-heavy: System Security Plan, narratives, supporting artifacts
- Independent 3PAO assessment and agency sponsor review
- Ongoing obligation after authorization: monthly continuous monitoring deliverables

What's coming - FedRAMP 20x

Sponsors and ATO

- ATO (Authority to Operate): a federal agency's formal decision that your service's risk is acceptable for its use
- Sponsor: the federal agency that partners with you, reviews your package, and grants the first ATO
- Under Rev5, you need an agency sponsor to begin

Agenda

1	Why FedRAMP	Motivation behind starting the project
2	**How it came to be**	General timeline and milestones
3	General k8s challenges	Nodes hardening and other requirements
4	Vulnerabilities	Scanning and patching vulnerabilities
5	Deployments	Dealing with unavailable external dependencies on the example of deployments

~~How the universe was born?~~ Why FedRAMP?

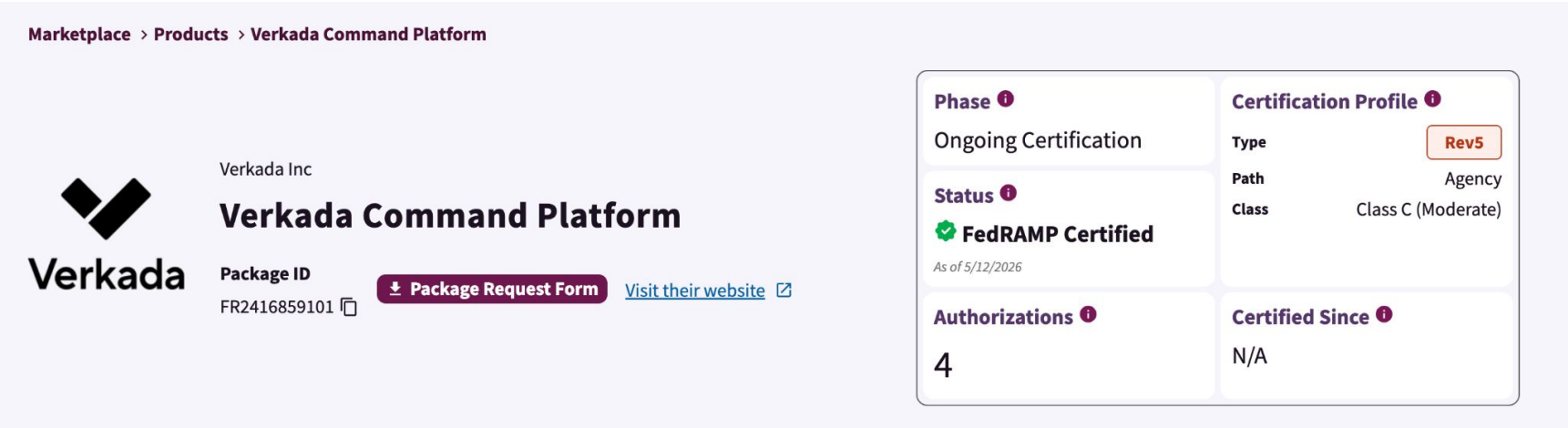
- We have great product (obviously I am biased)
- The target market had significant potential to justify this investment of time and resources
- We could afford to spend this time and resources (both monetary costs and engineering time)

Timeline



Q1 2023	Q2-Q3 2023	Q4 2023 - Q3 2024	Q4 2024	Q4 2025	May 2026
Q1 2204 **Preparations** External gaps analysis Decide on building environment ourselves and use vendor for consultin and documentation	**Initial Setup** Initial AWS setup, setup of dev infrastructure and EKS - following the same stack we used for non gov environment	**Setting up production** Setup of prod infrastructure and EKS Bringing up microservices in production	**Audit for FedRAMP Ready** Officially achieved status in **July 2025**	**Audit for FedRAMP Moderate**	**FedRAMP Certified (Moderate) status for Command hosted in AWS GovCloud**

Achievement unlocked (but work continues)



Agenda

1	Why FedRAMP	Motivation behind starting the project
2	How it came to be	General timeline and milestones
3	**General k8s challenges**	Nodes hardening and other requirements
4	Vulnerabilities	Scanning and patching vulnerabilities
5	Deployments	Dealing with unavailable external dependencies on the example of deployments

Challenges for making Kubernetes (EKS) cluster satisfy FedRAMP standards



Hardening node images



Compliant traffic encryption



Compliant cryptography in containers



Much more rigorous vulnerabilities scanning and reporting

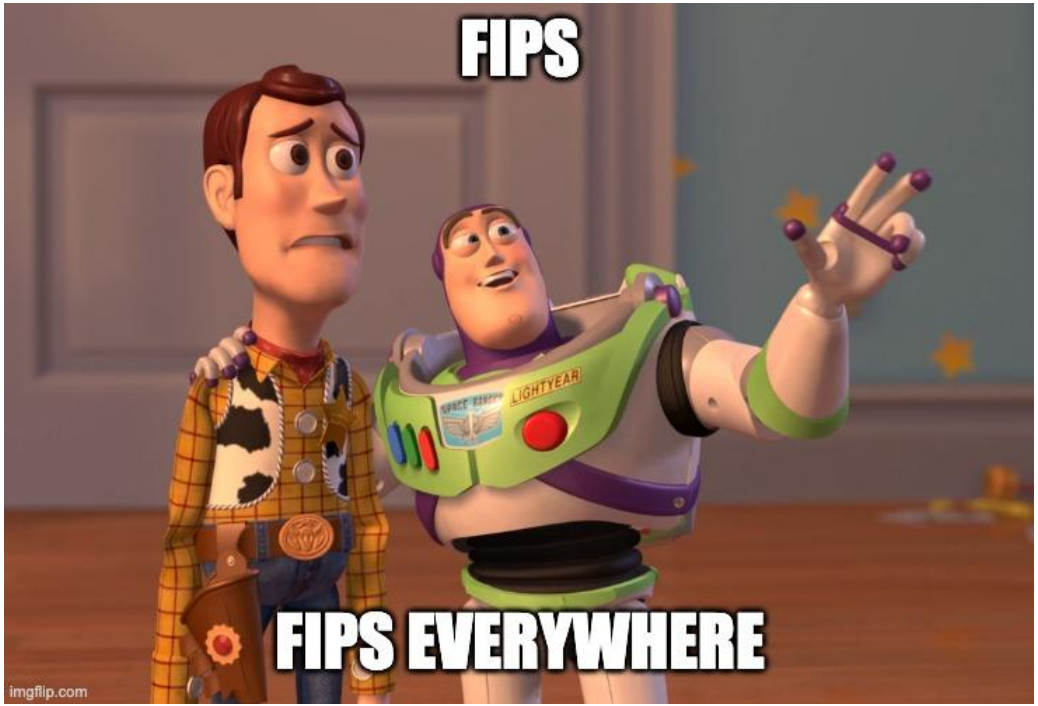


Unavailability of some fundamental external dependencies

FIPS

- FIPS = Federal Information Processing Standards, published by NIST
- The one that matters here is FIPS 140-3: security requirements for cryptographic modules (libraries, hardware, kernel crypto)
- Validated by accredited labs under NIST/CCCS's CMVP program; each module gets a certificate for a specific version and build
- Validated modules run self-tests at startup, use approved algorithms and random number generation, and block disallowed algorithms
- Why FedRAMP cares: cryptography must use FIPS-validated or NSA-approved modules

FIPS – short version



Nodes hardening

- We build custom node images:
  - STIG (Security Technical Implementation Guide) hardened build. Includes using FIPS mode
  - Additional monitoring software to satisfy our reporting and security requirements
- EC2 image builder is configured via Terraform
- Bonus: we switched to custom build for our non-gov environment clusters later as well and node startup time went from 10 min to 2 min

Compliant traffic encryption - Istio with mTLS

- Requirement is for all data to be encrypted in rest and in transit with (you guessed it) FIPS verified library/tool
- At the time we were using partial Isito service mesh in our non-gov cloud clusters
- Used full service mesh from the start and switched to ambient immediately when it became available
- Istio enforces mTLS
- Needed images with a magic word - FIPS
- Bonus - rolled out ambient later in our non-gov environment

Compliant cryptography in containers

- Required using FIPS validated cryptographic libraries
- We use monorepo, updated core libraries for everything and added some runtime switches
- Side note - firmware also requires FIPS libraries but it’s a story for another time

Agenda

1	Why FedRAMP	Motivation behind starting the project
2	How it came to be	General timeline and milestones
3	General k8s challenges	Nodes hardening and other requirements
4	**Vulnerabilities**	Scanning and patching vulnerabilities
5	Deployments	Dealing with unavailable external dependencies on the example of deployments

Vulnerabilities scanning and reporting

- Strict rules defined by FedRAMP program (ConMon - Continuous Monitoring) - every month we have a meeting with our sponsors (4 agencies) where we need to report alongside other things all overdue vulnerabilities and ask for any new exemptions
- We run around 70 different open source containers in our gov cloud clusters. All of those require constant patching to the latest versions
- Security team handles the reporting - combination of many management tools including spreadsheets to keep bureaucracy alive

Vulnerabilities patching

- This was the source of biggest operational load. We had multiple “war room” patching episodes due to various external factors (e.g., changed exemptions criteria).
- We obviously want to minimize the load
- Big limitations - it is not allowed to autodeploy in production environment due to the regulations

Vulnerabilities patching

- Switched to a vendor that builds open source images with 0 or small amount of vulnerabilities.
- Use Renovate project to automatically update dependencies in the repository. Treat this as a first class project.
- Build the processes to review/merge/deploy those changes proactively

Agenda

1	Why FedRAMP	Motivation behind starting the project
2	How it came to be	General timeline and milestones
3	General k8s challenges	Nodes hardening and other requirements
4	Vulnerabilities	Scanning and patching vulnerabilities
5	**Deployments**	Dealing with unavailable external dependencies on the example of deployments

Deployments

- Outside GovCloud, we use ArgoCD with Github repository for GitOps
- The problem - Github is not FedRAMP Moderate.
- The compliant technical solution was relatively stable
- The developer experience journey was way more bumpy

Deployments - implementation

- Use AWS CodeCommit in GovCloud as source of truth for ArgoCD
- Use Github for developer interactions
- Triggers and a service to mirror deployments between Github and CodeCommit
- The access to gov environment, ArgoCD in gov cloud clusters, CodeCommit is derived from requirements (US citizens and green card holders only, need to go through additional one time approval process)

Deployment experience v0

- Everything was done by the few people on dedicated team (during environment setup and initial phase)
- Separate tooling from our non-gov environment - less user friendly
- “Autodeploy” replaced by weekly updates

Deployment experience v1

- Tooling open to everyone with access to gov environment (slowly expanding pool of people)
- Still separate tooling from our non-gov environment - slightly more user friendly, now even with error handling
- Still weekly updates as poor man’s “autodeploy”
- Worked, but combined with v0 set the wrong expectation about deployments in gov - that they are happening outside of the usual development lifecycle
- We had several delayed feature releases and bug fixes due to missed deployments or dependencies

Deployment experience v2

- Everyone can trigger and see states of deployments with the same tooling as non-gov environment
- Requires approval from people onboarded to special reviewer group which expanded to include at least 1-2 person on each team if possible
- Standard autodeploy system generates PRs that now wait for team's approvals, they are in charge of their own deployments
- Successfully put ownership back where it belongs - people will do the right thing with good tooling and clear process

Summary

- It’s a long journey that required both technical adjustments and extensive documentation
- Delegating compliance when possible worked well for us
- At the same time many external vendors that we use non-gov environment are not available
- Consider how to handle development lifecycle, oncall, support, etc. based on your teams’ compositions with regards to govcloud access

Thank you!