**Plan outline**Absolutely — here's a strong set of **4–5 suggested questions** along with the **kind of answers, supporting material, or anecdotes** you might want to research to bring each one to life. These are designed to be engaging, informative, and spark discussion while subtly showcasing your expertise.

**1. What does "security best practice" *actually* mean in a cloud-native world?**

**Key ideas to research/discuss:**

* Contrast traditional (on-prem) security practices vs. cloud-native ones.
* The shift from perimeter-based to identity- and workload-based security models.
* Emphasize "shared responsibility" — how much is *really* on the customer in AWS, Azure, GCP?
* Anecdote: Example of a misconfiguration in cloud storage (e.g., open S3 buckets) that led to a breach — to show how "basic hygiene" still matters.

**2. CNAPP sounds like another acronym. What problem is it actually solving?**

**Key ideas to research/discuss:**

* Explain what CNAPP (Cloud-Native Application Protection Platform) really is — why Gartner coined it, what components it includes (CSPM, CWPP, CIEM, etc.).
* Talk about the sprawl of cloud tooling — companies often run **15–20+ tools** just for security.
* Discuss the pain of disconnected findings from CSPM, VM, container security, etc.
* Consider referencing a Gartner or IDC stat about tool fatigue or alert overload.

**3. Why is vulnerability management so much harder in the cloud than on-prem?**

**Key ideas to research/discuss:**

* The ephemerality of cloud assets — VMs, containers spin up/down in minutes.
* Traditional VM tools rely on IP addresses or agents — often not suited for serverless or containers.
* Highlight the challenge of **contextual risk**: not every vuln is equal — public exposure, privileges, asset criticality all change risk posture.
* Anecdote: Share a (sanitized) customer story where contextual prioritization helped avoid a "false high" or revealed a true critical.

**4. How do I get visibility into my actual cloud risk? Isn’t it all too dynamic to track?**

**Key ideas to research/discuss:**

* The need for **real-time or near-real-time inventory** in multi-cloud.
* Mention the value of integrating with native APIs (AWS Config, Azure Resource Graph, GCP Security Command Center).
* Discuss the value of normalized tagging, asset classification, and using posture data alongside vulnerabilities.
* Stat to research: Percentage of cloud assets typically untagged in enterprise environments.

**5. Where should a team start if they’re just beginning cloud VM?**

**Key ideas to research/discuss:**

* Start with visibility — you can’t protect what you can’t see.
* Prioritize coverage of internet-facing assets and critical workloads.
* Consider managed VM solutions that support agentless scanning as well as runtime analysis.
* Research frameworks to reference: CIS Benchmarks, NIST CSF, AWS Well-Architected Security Pillar.
* Optional anecdote: A small team that made big gains quickly by focusing first on just one cloud account or business unit.

**Question 1: What does ‘security best practice’ actually mean in a cloud-native world?**

**🎯 Target time: ~5–6 minutes**

**🧑‍💼 Host (kickoff):**

"Let's kick off with a big one. We hear the phrase *'security best practice'* constantly — but in a cloud-native world, where things are spinning up and down every second, does that concept still hold the same weight? Or has the definition changed?"

**👨‍🏫 Expert:**

"That’s the right place to start. The truth is, traditional notions of 'best practice' were designed for static environments. Think of it like managing a fixed set of servers behind a firewall — it was all about locking down the perimeter, patching systems, segmenting networks.

But cloud has changed all of that. And at the heart of this shift is something people often overlook: **NIST’s principle of Rapid Elasticity.** It’s the idea that cloud resources can scale dynamically — they appear and disappear in minutes, not weeks.

That makes the entire attack surface fluid and ephemeral — and the old ‘scan it and forget it’ model breaks completely."

**🔹 Beat 1: The real challenge — Rapid Elasticity (NIST 800-145)**

“Rapid Elasticity isn’t just a convenience — it’s the root of why cloud security is fundamentally hard. If your infrastructure is constantly shifting, you need visibility that moves at that same speed.

It’s not just ‘protect your servers’ anymore — it’s *track everything that exists right now*, because tomorrow’s environment won’t look like today’s.”

"And that’s exactly what contributed to the **Capital One breach**. A misconfigured S3 bucket and an over-permissive IAM role — both created as part of automated infrastructure provisioning — were exploited.

It wasn’t some zero-day vulnerability. It was a visibility and configuration lapse that was spun up and missed in real time."

**🔹 Beat 2: Shared Responsibility — Often Misunderstood**

“Then you’ve got the **shared responsibility model** — which sounds simple in theory, but causes a lot of confusion in practice.

Cloud providers secure the infrastructure — the physical hardware, the network, the hypervisor. But *you* are responsible for your data, your apps, your identities, your configurations."

“It’s like renting an apartment. The landlord locks the building — but it’s on you to lock your door and not give your key to everyone on the street.”

"Gartner said it clearly: **99% of cloud security failures will be the customer’s fault**. That usually comes down to misconfigurations, like public buckets or open ports — because teams assume ‘the cloud provider has it covered.’ They don’t."

**🔹 Beat 3: Identity is the new perimeter**

"In a traditional data center, we had a moat-and-wall model: build a hard perimeter, and trust what’s inside. That model doesn’t translate to cloud — because in cloud, there *is* no perimeter.

So today, **identity is the new perimeter**. If someone gets access to an over-permissive IAM role, or an API key, they don’t need to 'hack in.' They just walk in."

“That’s what happened in the **Uber breach** — credentials were compromised, and once inside, the attacker had access to internal systems that weren’t protected with least privilege. They didn’t breach the infrastructure — they *used identity*.”

**Tactical best practices to highlight:**

* Least privilege everywhere
* Enforce MFA across all identities (including service accounts)
* Regular access reviews and automatic key rotation
* Monitor for privilege escalation paths

**🔹 Beat 4: Best practice = Continuous Validation**

“And finally, this all leads us to the modern definition of cloud security best practice:  
It’s not a checklist. It’s a **continuous validation process**.”

“Because cloud environments are constantly evolving, *your visibility must be continuous*, not periodic. You can’t scan once a week and hope you’re covered. You need to know what’s happening in near real time.”

Here’s what that looks like:

**✅ You need continuous visibility into:**

* **Asset Inventory:** Know what resources exist at any given moment — VMs, containers, buckets, databases, serverless functions, etc.
* **Configuration Management:** Validate settings like firewall rules, encryption status, and network exposure — and catch drift immediately.
* **Access & Identity:** Understand who (or what) can access what. Detect and correct excessive permissions before they’re exploited.
* **Exposure & Threat Detection:** Watch for publicly exposed assets, anomalous behavior, or lateral movement — as it’s happening.

“Modern best practice means building a system that can *see, evaluate, and respond* continuously — because if you wait for a scheduled scan or manual audit, you're already behind.”

**🧑‍💼 Host (wraps the section):**

"That really reframes it. Best practice today is more about living systems — real-time feedback loops — not static rules.

And what stuck with me? *Identity is the new perimeter.* That’s a game changer. It explains why misconfigurations and overly broad roles keep leading to breaches."

**💬 Optional audience engagement prompts:**

* **Poll:** How often do you review IAM permissions in your cloud accounts?
* **Chat prompt:** Have you ever discovered a 'temporary' IAM role or S3 bucket that turned out to be wide open months later?

**🎙️ Question 2: “CNAPP sounds like another acronym. What problem is it actually solving?”**

**🟠 Beat 1: The Problem — Fragmented Tooling and Cloud Complexity**

“Most cloud security teams today are working with a dozen or more disconnected tools — one for cloud posture, one for workload scanning, another for permissions, another for runtime, and so on.”

“Each tool might be good at what it does, but none of them talk to each other. So teams get flooded with alerts — most of which lack context — and they’re left stitching together clues manually to understand what’s actually important.”

“Meanwhile, developers are shipping faster, infrastructure is ephemeral, and identity has become the new perimeter. The old model just doesn’t scale.”

**🟦 Beat 2: What CNAPP Is — A Unified Platform Across the Cloud App Lifecycle**

“This is where CNAPP comes in. It’s not just a category — it’s a consolidation of the key capabilities teams already use, but delivered as a single, integrated platform.”

“That includes CSPM for misconfigurations, CWPP for workload protection, CIEM for identity risk, IaC security for shift-left scanning, and often runtime threat detection like CDR or API scanning.”

“The idea is simple: if cloud-native applications are built and run in one continuous system, your security should work the same way. CNAPP gives you visibility from code to cloud, across posture, identity, runtime, and more — in one place.”

**📊 Slide 1 (Supports Beat 2): “What Makes Up a CNAPP?”**

**Visual:**  
A **layered platform diagram** or **lifecycle flow**, showing:

* Developer pipeline → IaC scanning
* Cloud deployment → CSPM, CIEM
* Runtime workloads → CWPP, CDR
* Identity context across all stages

**Caption idea:**

“CNAPP brings together core security pillars — posture, workload, identity, and runtime — into a single, unified platform.”

**🟩 Beat 3: Why CNAPP Matters — Risk Context and Actionability**

“But unifying tools is just the start. The real power of CNAPP is what happens when those signals are connected.”

“Let’s say a container is running a known CVE, and it’s exposed via a misconfigured S3 bucket — and the IAM role tied to it has excessive permissions. Traditional tools might surface those as separate alerts, but CNAPP correlates them into a single risk chain.”

“That’s the difference: instead of noise, you get context. You know which risks are truly exploitable, which identities can reach them, and what action to take.”

“Then you can automate the fix — push a remediation to the cloud provider, create a Jira ticket, or block suspicious runtime behavior. That’s how teams move faster *and* reduce risk.”

**📊 Slide 2 (Supports Beat 3): “How CNAPP Turns Noise into Risk-Based Action”**

**Visual:**  
A **left-to-right flow diagram**:

* Inputs: Disconnected alerts (CVE, misconfig, IAM risk)
* Middle: CNAPP engine → correlates into risk path, prioritizes
* Outputs:
  + Auto-remediation
  + Dev pipeline fix
  + Runtime protection

**Caption idea:**

“CNAPP connects the dots — across config, code, identity, and runtime — and helps teams fix what actually matters.”

**🔴 Beat 4: Without CNAPP — More Work, Less Clarity, Higher Risk**

“Without CNAPP, you’re left with fragmented alerts, duplicated effort, and slower time to response.”

“You miss the connections between posture, vulnerability, and identity. You burn cycles triaging noise. And the most critical risks — the ones that *can actually be exploited* — often go unnoticed until it’s too late.”

“Security becomes reactive, not preventative. And that’s a gap most teams can’t afford.”

**✅ Final Summary (optional closer or TL;DR)**

“So CNAPP isn’t just another acronym — it’s a response to the complexity of cloud-native environments. It unifies core tools, gives teams context, and helps them fix what matters most — before it becomes a breach.”

**Question 3: “Why is vulnerability management so much harder in the cloud than on-prem?”**

**🟠 Beat 1: The Cloud Broke the Old Vulnerability Model**

“In the on-prem world, vulnerability management was predictable. You had known hosts, fixed IPs, long-lived infrastructure, and agent-based tools that ran on a quarterly or monthly cycle.”

“That model just doesn’t work in the cloud. You’ve got assets spinning up and down constantly — containers that live for minutes, functions that only exist when invoked, and infrastructure defined in code and deployed on demand.”

“You can’t rely on periodic scans or static tracking anymore. The speed and structure of the cloud has outpaced the traditional model.”

**🟦 Beat 2: Visibility Is Fragmented — and Time-Sensitive**

“Cloud environments are dynamic and multi-layered. You've got workloads, images, packages, identities, services, infrastructure-as-code — and each layer introduces different risks.”

“You might still run agents — and they can be useful — but they won’t help you detect a vulnerability in a container image stored in a registry, or a Lambda function that hasn't been invoked yet, or an exposed S3 bucket with public access. You need visibility into *everything* — not just what’s running, but what *could run*.”

“And speed matters. If your scanner sees an asset hours after it was deployed, that might be too late.”

**📊 Slide 1 (Supports Beat 2): “The Visibility Gaps in Cloud Vulnerability Scanning”**

**Visual Idea:**  
A **grid or heatmap** showing scanning coverage across different cloud asset types:

|  | **Agent-Based Tools** | **IP Scanners** | **Cloud-Native/API** |
| --- | --- | --- | --- |
| Long-lived VMs | ✅ Full | ✅ Full | ✅ Full |
| Containers (running) | ⚠️ Partial | ❌ Missed | ✅ Full |
| Container Images | ❌ Missed | ❌ Missed | ✅ Full |
| Serverless (e.g. Lambda) | ❌ Missed | ❌ Missed | ✅ Full |
| Cloud Services (S3, RDS) | ❌ Missed | ❌ Missed | ✅ Full |

**Caption:**

“In the cloud, most scanning tools only see part of the picture. You need layered, real-time visibility — not just agents on hosts.”

**🟩 Beat 3: Not Every Vulnerability Matters — Context Is Key**

“Here’s the real challenge: cloud teams don’t just need to find vulnerabilities — they need to understand which ones are *actually dangerous*.”

“A ‘critical’ CVE on an isolated dev VM with no access to anything is probably noise. But a medium CVE on an internet-facing container that has an overprivileged IAM role and access to sensitive data — *that’s* urgent.”

“That’s where context matters: is the asset exposed? Is it reachable from the internet? Who can access it? What does it have access to? Is it active right now?”

“Wiz correlates all of this — vulnerabilities, cloud posture, exposure, identity, secrets, runtime activity — into a single risk graph. That gives you *true* prioritization.”

**📊 Slide 2 (Supports Beat 3): “What Makes a Vulnerability *Actually* Dangerous?”**

**Visual Idea:**  
A **risk chain diagram** showing how a “medium” CVE can lead to compromise:

* **Node 1**: Container with a medium-severity CVE
* **Node 2**: Public internet exposure
* **Node 3**: Overprivileged IAM role
* **Node 4**: Access to internal database

Arrows show the attacker path → with short callouts:

* “Reachable from internet”
* “Privilege escalation”
* “Lateral movement”

**Caption:**

“Severity alone doesn’t define risk — it’s exposure, privilege, and blast radius that matter.”

**🔴 Beat 4: Customer Story — Context Changed Everything**

“We had a customer in the fintech space who was drowning in thousands of vulnerability alerts — their scanners flagged everything critical, without context.”

“But once they used Wiz, they saw that many of those criticals were on isolated, non-production assets. Meanwhile, a few medium CVEs surfaced at the top — they were on active containers, exposed to the internet, with access to customer data via misconfigured IAM.”

“That shift — from volume to *risk-based prioritization* — changed how they worked. Instead of reacting to noise, they focused on what actually mattered.”

**🗣️ Over to You (Poll + Chat Prompt Slide)**

**🔹 Poll:**

**Title:**  
**“Where do your biggest visibility gaps exist today?”**

**Options:**

* Containers or container images
* Serverless functions (e.g. Lambda)
* Cloud services (e.g. S3, RDS)
* IAM and privilege configurations
* We’re not sure — we just react to alerts

**🔹 Chat Prompt:**

**Question:**  
**“Ever had a ‘critical’ vuln alert that turned out to be nothing… or a ‘medium’ that turned into a fire drill?”**

Drop a 🚨 in the chat if you’ve seen misprioritization slow your team down — or if context helped you catch a real issue early.

(Bonus: feel free to share a quick example, like “dev container w/ prod secrets.”)