📝 CHECKOV

Securing Infrastructure as Code before it’s deployed

1. What is Checkov?

- **Checkov is a static code analysis tool commonly used to scan Infrastructure as Code**
- It checks **security & compliance policies**
- Works on:
  - Terraform
  - Kubernetes YAML
  - Helm
  - CloudFormation

It **reads code only** (does NOT deploy anything)

> Checkov is a static IaC security scanner that enforces resource-level (CKV) and architecture-level (CKV2) policies before infrastructure is deployed

2. What does Static Code Analysis mean?

- **Analyzing code without executing it**
- No AWS resources are created
- No runtime testing

Example:

- S3 marked `public-read` → ❌ flagged before deploy

This is called **Shift-Left Security**

3. What does Checkov check?

- Security issues (public access, open ports)
- Best practices (encryption, logging)
- Compliance rules

Examples:

- S3 bucket public?
- RDS encrypted?

- Security group open to `0.0.0.0/0`?
- EC2 reachable from internet?

4. How to install & run Checkov

Install

```
pip install checkov
```

Run scan

```
checkov -d .
```

This scans **all IaC files in the folder**

5. How Checkov decides which policies to run

- Checkov **auto-detects file types**
- Applies **only relevant policies**

File type	Policies applied
`.tf`	Terraform policies
`k8s yaml`	Kubernetes policies
`helm`	Helm policies

No need to download policies manually

6. Do all 750+ policies run?

- Yes, **by default**
- But **only those relevant to your code**

Example:

- If no Kubernetes YAML → k8s policies won't run

7. What is CKV?

- **CKV = Resource-level policies**
- Checks **attributes inside one resource**
- Written in **Python (** `.py` **)**

Examples:

- S3 public access
- RDS encrypted
- EC2 has public IP
- SG allows `0.0.0.0/0`

CKV checks **each resource independently**

8. What is CKV2?

- **CKV2 = Graph / relationship-based policies**
- Checks **how resources are connected**
- Written in **YAML (** `.yaml` **)**

Examples:

- EC2 reachable from internet
- RDS in private subnet with no IGW
- EBS attached to EC2 and encrypted
- ALB protected by WAF

CKV2 checks **architecture, not just config**

9. CKV vs CKV2 (easy rule)

Question	Use
Is resource configured correctly?	CKV
Is resource reachable / connected securely?	CKV2
Needs “attached to / associated with”?	CKV2

Key difference (IMPORTANT)

- **CKV checks many resources of the same type**
- **CKV2 checks relationships between different resources**

🔍 How to identify CKV vs CKV2

Indicator	Meaning
`.py` file	CKV
`.yaml` file	CKV2
Mentions “attached / connected / reachable”	CKV2

10. Can we run CKV and CKV2 together?

YES (Recommended & Standard)

Example:

```
checkov -d . \
--check CKV_AWS_20 \
--check CKV_AWS_130 \
--check CKV2_AWS_18
```

CKV + CKV2 = **complete security coverage**

11. Ways to execute Checkov

- Local CLI
- CI/CD (GitHub Actions, Jenkins)
- Config file (`.checkov.yaml`)

Example config:

```
check:
  - CKV_AWS_20
  - CKV_AWS_130
  - CKV2_AWS_18
```

12. Relation to other tools

Tool	Type
Checkov	Static
tfsec	Static
OPA (for IaC)	Static
AWS Config	Dynamic

13. Baseline vs Gating (CI/CD concept)

- **Baseline**
  - Shows all issues
  - ❌ Pipeline does NOT fail
  - Used when introducing Checkov to existing code
  - Also called **soft-fail**

- **Gating**
    - Blocks PR / pipeline on violations
    - ❌ Pipeline fails
    - Used for enforcing security
    - Also called **hard-fail**

These are **CI/CD concepts**, not Checkov keywords

14. Soft-fail vs Hard-fail (mapping)

Term	Meaning
Soft-fail	Report issues only
Hard-fail	Fail pipeline

Examples:

```
# Soft-fail (baseline)
checkov -d . || true

# Hard-fail (gating)
checkov -d . --severity-threshold HIGH
```

15. Severity levels in Checkov

Each policy has a severity:

- LOW
- MEDIUM
- HIGH
- CRITICAL

Severity is **per policy**, not CKV / CKV2

16. Where severity is defined (IMPORTANT)

- ❌ NOT inside `.py` (CKV) code
- ❌ NOT inside `.yaml` (CKV2) code
- ✅ Defined in **policy metadata registry**
- Shown in:
    - CLI output

    - JSON reports
    - Official policy documentation

17. `-severity-threshold` meaning

```
checkov -d . --severity-threshold HIGH
```

What it does:

- Runs **ALL applicable policies**
- Reports **ALL severities**
- ❌ Pipeline fails only if:
    - HIGH or CRITICAL found

❗ It does **NOT** mean "run only HIGH policies"

18. Can we run only HIGH severity policies?

- ❌ No direct flag like `-run-only-high`
- Severity is resolved **after execution**

What teams do instead:

- Identify HIGH/CRITICAL policy IDs
- Run them explicitly using `-check`

---

19. Reporting HIGH/CRITICAL only (no pipeline failure)

```
checkov -d . --severity-threshold HIGH || true
```

or in GitHub Actions:

```
continue-on-error: true
```

Used for **visibility-only reporting**

20. Compliance frameworks (Beginner meaning)

- Compliance = **security rulebooks**
- Created by security / regulatory bodies
- Required for audits & customer trust

Common frameworks:

- CIS
- SOC2
- ISO 27001
- PCI DSS

21. How Checkov supports compliance

- Each policy maps to:
    - CIS controls
    - SOC2 controls
    - ISO clauses (some policies)
- Reports clearly show mappings

Helps prove:

> "These controls are enforced"

(Checkov ≠ certification, only evidence)

22. Terraform plan scanning

Why scan plan?

- Variables resolved
- Conditionals evaluated
- More accurate than `.tf` only

Commands:

```
terraform plan -out=tfplan
terraform show -json tfplan | checkov -f -
```

MNC practice:

- PR stage → scan `.tf`
- Pre-prod / prod → scan plan
- Often **both**

23. `.checkov.yaml` file (central config)

- Central place for Checkov settings
- Avoids long CLI commands
- Used by local & CI/CD

Example:

```
framework:
  - terraform

check:
  - CKV_AWS_20
  - CKV_AWS_130
  - CKV2_AWS_18

skip-check:
  - CKV_AWS_1

severity-threshold: HIGH
```

Meaning:

- Only listed policies run
- One policy skipped
- Pipeline fails only on HIGH+

24. `check` , `skip-check` , `severity-threshold` (clear roles)

Setting	Purpose
`check`	Which policies run
`skip-check`	Which policies ignored
`severity-threshold`	When pipeline fails

25. Suppressing / Skipping policies (why needed)

Even with selected rules:

- False positives happen
- Architecture handled differently
- Dev vs prod differences

Example:

```
# checkov:skip=CKV_AWS_20 reason: access controlled via IAM policy
```

Skipping = **documented exception**, not ignoring security

26. False positives

- Tool flags an issue
- Real risk does NOT exist

Example:

- DB flagged public
- But no IGW + VPN-only access

---

27. Risk acceptance

- Security team accepts known risk
- Reason documented
- Temporary or business-driven

Auditors expect:

- Reason
- Tracking (Jira / ticket)
- Review process

28. Official policy documentation (IMPORTANT)

Use official policy pages to check:

- Severity
- Category
- Compliance mapping

Sources:

- Checkov Policy Index (IDs & descriptions)
- Prisma Cloud / Bridgecrew policy reference (severity & compliance)

Summary / Ending Note

To summarize, **Checkov** is a static analysis tool that helps identify security, compliance, and best-practice issues in Infrastructure as Code before anything is deployed. It works by scanning IaC files such as Terraform, Kubernetes manifests, and Helm charts and applying built-in policies automatically based on the type of code being analyzed. Resource-level checks (**CKV**) focus on whether individual resources are configured correctly, while graph-based checks (**CKV2**) evaluate how multiple resources are connected and whether the overall architecture is secure.

Severity is not defined inside policy code itself but is maintained as metadata, which allows teams to control enforcement through CI/CD pipelines using severity thresholds. This enables different workflows such as **baseline (visibility-only)** checks during early stages and **gating (enforcement)** checks for higher environments like production. Additionally, Checkov maps many policies to common compliance frameworks such as CIS, SOC2, and ISO, making it useful not only for security but also for audit readiness and governance discussions.

For deeper reference on policies, severity levels, and compliance mappings, the following official documentation pages can be used:

- **Prisma Cloud / Bridgecrew – AWS Policy Reference (severity & compliance mapping):**

  https://docs.prismacloud.io/en/enterprise-edition/policy-reference/aws-policies/aws-policies

- **Checkov Official Terraform Policy Index:**

  https://www.checkov.io/5.Policy%20Index/terraform.html

Together, these resources provide the authoritative source of truth for understanding how Checkov policies are defined, classified, and mapped to real-world security and compliance requirements.

Name: Abhinay Yalla
Role: DevOps Engineer