**Understanding the 4Cs of Cloud Native Security**

**Code, Container, Cluster, and Cloud**

Cloud-native security is built in layers, just like an onion. There are **4 key layers**, often called the **4Cs**:

* **Code** (Innermost layer)
* **Container**
* **Cluster**
* **Cloud** (Outermost layer)

Let’s break each one down in very simple terms.

**1. Code – The Core of the Application**

**What is Code?**

Code is the heart of your application. It’s the actual instructions that tell your software what to do.

**Code Quality vs Code Security**

* **Code Quality** is about how well-written and efficient your code is.
* **Code Security** is about making sure the code can’t be misused or attacked by hackers.

**What is a Code Vulnerability?**

A **code vulnerability** is a mistake or flaw in the code.

* Hackers can use these flaws to attack the system.
* They might steal or delete your data by taking control of your app or device.

**What is Security Code Analysis?**

This means **checking your code** to find any weaknesses or flaws.  
It can be done:

* **Automatically** (using tools), or
* **Manually** (by humans reviewing the code).

This process helps catch logic errors and possible attack points.

**Tools to Secure Code**

* **SAST** – Static Application Security Testing (checks the code without running it).
* **DAST** – Dynamic Application Security Testing (checks the app while it’s running).
* **IAST** – Interactive Application Security Testing (combines both).

**How to Secure Your Code**

* Use scanning tools (like SAST, DAST, IAST).
* **Limit port ranges** – don’t leave unnecessary ports open.
* **Scan third-party libraries** (these are pieces of code from others that your app uses).

**Who’s Responsible?**

* Mostly the **developers**.

**Examples**

* Regularly scan your code for weaknesses.
* Follow **secure coding practices**, like:
  + Validating input from users (to prevent things like SQL Injection or XSS).
* Keep updating your code to fix any security issues found.

**2. Container – The Next Layer of Security**

**What is a Container?**

Think of a container like a sealed box that holds your code and everything it needs to run. It’s portable and runs the same everywhere.

**What is Docker Container Security?**

Securing the container means making sure:

* The image (the “box”) has no known bugs or issues.
* The container runs safely and doesn’t expose your app.

**How to Secure a Container**

* Scan the container **and the operating system** inside it while building the Docker image.
* Use strong **IAM (Identity and Access Management)**.
* Always create users with **only the access they need** (least privilege).

**Who’s Responsible?**

* **Developers** and **Security Teams** work together.

**Examples**

* Regularly scan containers for vulnerabilities.
* Use **strong identity rules**.
* Create users with minimal access.

**3. Cluster – The Management Layer**

**What is a Cluster?**

A cluster is a group of machines (or nodes) that run your containers and manage how they work together. This is what Kubernetes handles.

**Cluster Security Focus**

* Protect the **software** that controls the app and its resources.

**How to Secure a Cluster**

* Use **RBAC (Role-Based Access Control)** – this means controlling **who can do what** inside the cluster.
* Regularly update Kubernetes and its control components with the latest **security patches**.

**Who’s Responsible?**

* **Kubernetes Admins** (CKA, CKS)
* **Security Teams** (CKS holders)

**4. Cloud – The Infrastructure Layer**

**What is the Cloud?**

The cloud is the overall system where everything runs – like AWS, GCP, Azure, etc. This includes physical machines, virtual machines, networks, and more.

**Cloud Security Focus**

* Protect the **infrastructure** and **communication channels**.

**Who’s Responsible?**

* **Inside the company**: Cloud Admins
* **Outside the company**: Cloud Providers (AWS, Azure, GCP, etc.)

**Examples**

* **Separate virtual machines** so they don’t interfere with each other.
* Secure APIs with **authentication** (only allowed users can access).
* Use **encryption** to protect data during transfer.