### **What is SkyTransit?**

Imagine you have a gaming console at home (like a PlayStation) and you want to play an online game that’s stored on Amazon’s cloud (AWS). The problem is, connecting your console to the cloud can be tricky and unsafe if you don’t do it right. SkyTransit is like a special, secure tunnel that helps your console connect to the cloud easily and safely.

### **Why Use SkyTransit?**

Normally, connecting your home console to the cloud takes a lot of steps, and you have to set up everything yourself, which can be messy and risky. SkyTransit does all the hard work for you, so you can connect quickly and safely using something called a "PrivateLink." It’s like a secret tunnel that only lets your console talk to the cloud, keeping everything private and secure.

### **How Does It Work? (The Picture Explained)**

The picture shows how SkyTransit sets up this connection:

* **Your Side (Desktops/Servers)**: This is your gaming console (or computer) at home or in an office.
* **Firewalls**: These are like security guards that protect your console from bad stuff on the internet.
* **AWS Direct Connect**: This is a special cable that connects your console to Amazon’s cloud (AWS).
* **Transit VPC**: Think of this as a checkpoint in the cloud that makes sure only the right stuff gets through.
* **VPC Endpoint (with AWS PrivateLink)**: This is the end of the tunnel—it connects directly to the game (or service) you want to use on AWS. It makes sure your connection is private and only talks to the right part of the cloud.
* **AWS Service**: This is the game or app you’re trying to use on AWS.

There’s also a note about something called a "Stubbed CNAME" (don’t worry about the fancy term)—it just means you can give the tunnel a nickname to make it easier to use.

### **Key Point**

The tunnel (SkyTransit) only lets your console talk to specific AWS services that are approved by your organization (called GS). It’s like saying, “You can only play this one game, nothing else,” to keep things safe.

So, SkyTransit is like a safe, easy bridge between your computer and Amazon’s cloud, so you can use apps or games there without worrying about security or complicated setup. Does that make sense?

### **Docker vs. Fast Track**

* **Docker** is like a lunchbox: It packs your app and everything it needs (like ingredients—code, libraries, settings) into a “container.” This container can run on any computer that has Docker, so your app works the same everywhere. Docker’s main job is to make your app portable and easy to run.
* **Fast Track**, on the other hand, is more like a full kitchen with rules: It’s a system that helps you build, test, and launch your app on Amazon’s cloud (AWS). It’s not just about packing your app (like Docker does)—it’s about the whole process of creating and managing the app, keeping it secure, and making sure it follows company rules.

### **What’s the Purpose of Fast Track?**

Fast Track helps you build apps on AWS faster and safer by giving you:

1. **A Guided Process**: Imagine you’re cooking in a kitchen where someone already set up the tools, recipes, and safety rules for you. Fast Track does that for building apps—it sets up a secure way to code, test, and launch your app on AWS.
2. **Safety Rules (Guardrails)**: It makes sure your app is secure and follows the company’s rules (like not leaving the stove on in the kitchen). For example, it checks for security problems and fixes them automatically or tells you to fix them.
3. **Two Work Areas**:
   * **Sandbox**: A practice area (like a mini-kitchen) where you can test your app without worrying about messing up.
   * **Production**: The real kitchen where your app goes live for people to use, with stricter rules to keep everything safe.
4. **Teamwork and Tools**: It lets you work with friends using tools like GitLab (a place to store your app’s code) and AWS tools to build and manage your app.

### **So, Is Fast Track Like Docker?**

Not really. Docker is just about packing and running your app in a container. Fast Track is about the whole process of building, testing, and launching your app on AWS, with extra focus on keeping things secure and following rules. You *could* use Docker inside Fast Track to run your app, but Fast Track does way more—it’s like a manager that guides you through the entire cooking process, not just the lunchbox that holds your food.

### 

### **What’s This About?**

This is about setting up stuff on Amazon’s cloud (called AWS) using a system called SkyFoundry. SkyFoundry helps you build and manage things on AWS, but you need to know a few basics first. Let’s explain it like you’re setting up a gaming server for your friends to play on!

### **AWS (Amazon Web Services)**

AWS is like a giant playground on the internet where you can build and run apps, games, or websites. SkyFoundry only works with AWS, so you need to learn a few things about AWS before you start building. The document gives you some topics to understand:

1. **Getting Started with AWS**: This is like reading the rulebook for the playground. It teaches you how to sign up and start using AWS.
2. **Private Cloud in AWS (VPC)**: Think of this as your own private corner in the playground where only you and your friends can play. It keeps your stuff separate and safe from others.
3. **Identity Access Management (IAM)**: This is like giving your friends a special pass to enter your corner of the playground. It decides who can come in and what they’re allowed to do (like only playing certain games).
4. **Key Management Service (KMS)**: This is like a lockbox for your secrets (like passwords or game codes). It keeps them safe so no one else can steal them.

### **Terraform**

Terraform is a tool that helps you build your playground setup automatically. Imagine you want to set up swings, slides, and a sandbox in your corner of the playground. Instead of building each one by hand, Terraform lets you write a plan (like a blueprint) and then builds everything for you.

* You’ll store this plan in a place called GitLab (like a notebook where you keep your ideas).
* Terraform will use that plan to set up your stuff on AWS.

**Important Note**: SkyFoundry doesn’t manage Terraform for you. It’s just the tool you use to build things. If you need help with Terraform, you have to ask a group called the Sky IaC team.

### 

### **What Does SkyFoundry Do to Help?**

SkyFoundry gives you ready-made blueprints (called Terraform modules) to make building easier. It’s like getting pre-made kits for your swings and slides instead of starting from scratch. These kits come with instructions to help you set up your playground the right way.

### **So, What Do You Need to Do?**

1. Learn the basics of AWS (the playground) using the topics they listed.
2. Use Terraform (the building tool) to set up your stuff on AWS.
3. Follow SkyFoundry’s ready-made blueprints to make building faster and easier.
4. If you get stuck with Terraform, ask the Sky IaC team for help.

### **What’s This About?**

Imagine you’re creating a game and want to put it on Amazon’s cloud (AWS) so your friends can play it online. SkyFoundry is a tool that helps you do this automatically using something called a GitLab pipeline. Think of it like a conveyor belt that takes your game, sets it up, and puts it on AWS for you.

### **What Does SkyFoundry Help With?**

SkyFoundry can set up two types of things on AWS for your game:

1. **AWS ECS**: This is like a shelf where your game can sit and run. It’s for games that are packed in containers (like a lunchbox—remember Docker from before?).
2. **AWS Lambda**: This is for smaller pieces of your game that only run when needed, like a mini-game that pops up when someone clicks a button.

You can also use something called **AWS SSM Parameter Store** to store secret info (like a password for your game) safely.

### **The Conveyor Belt (GitLab Pipeline) Explained**

The picture shows how SkyFoundry works like a conveyor belt to set up your game:

1. **You Start**: You (the developer) finish coding your game and send it to a place called GitLab (like a storage box for your game’s code).
2. **The Conveyor Belt Starts**: GitLab has a pipeline (the conveyor belt) that uses SkyFoundry to do the work for you.
3. **What Happens on the Belt**:
   * **For ECS (the shelf for your game)**:
     + SkyFoundry takes your game’s code from GitLab.
     + It updates the game’s settings (using AWS Parameter Store for secrets).
     + It packs your game into a container and puts it on the ECS shelf.
     + It also saves your game to a storage spot called S3 (like a backup folder).
     + Finally, it starts the game on ECS so your friends can play!
   * **For Lambda (the mini-game)**:
     + SkyFoundry takes the code for your mini-game.
     + It packs it up and saves it to S3.
     + It updates the mini-game on Lambda so it’s ready to run when someone clicks that button.

### **What Do You Need to Do First?**

Before you can use this conveyor belt:

1. You need to use **AWS GitLab** to store your game’s code (it’s like a special storage box Amazon provides).
2. You need to set up the playground (AWS) using SkyFoundry’s blueprints (called Terraform modules—like the ones we talked about before). This makes sure the shelf (ECS) or mini-game spot (Lambda) is ready.

### **Extra Info**

SkyFoundry has a tool called a “client” that helps you control this conveyor belt. It’s like a remote control for setting up your game. You can also check out examples of other games that use SkyFoundry to see how it works.