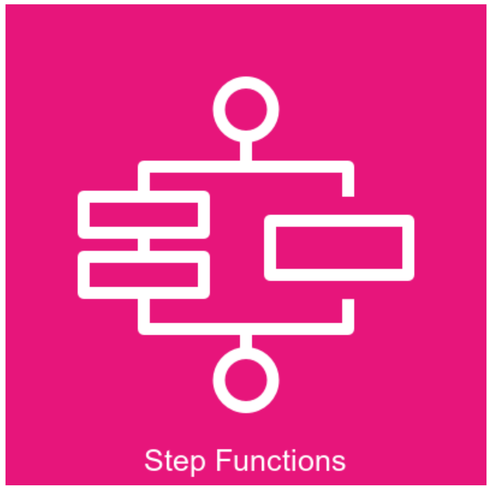
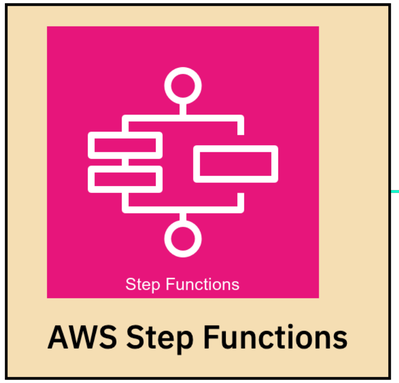
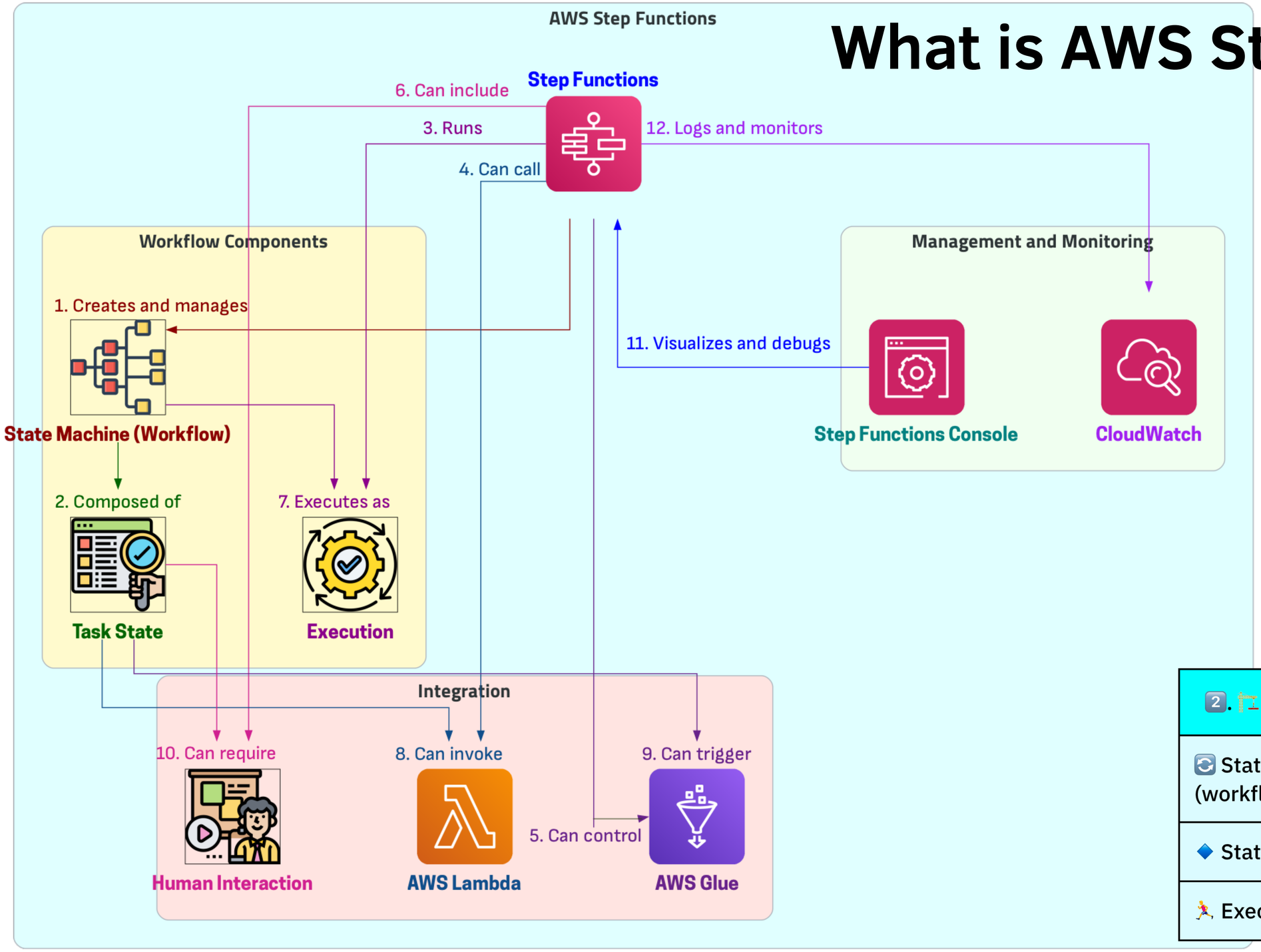


AWS Step Functions

Table of Contents



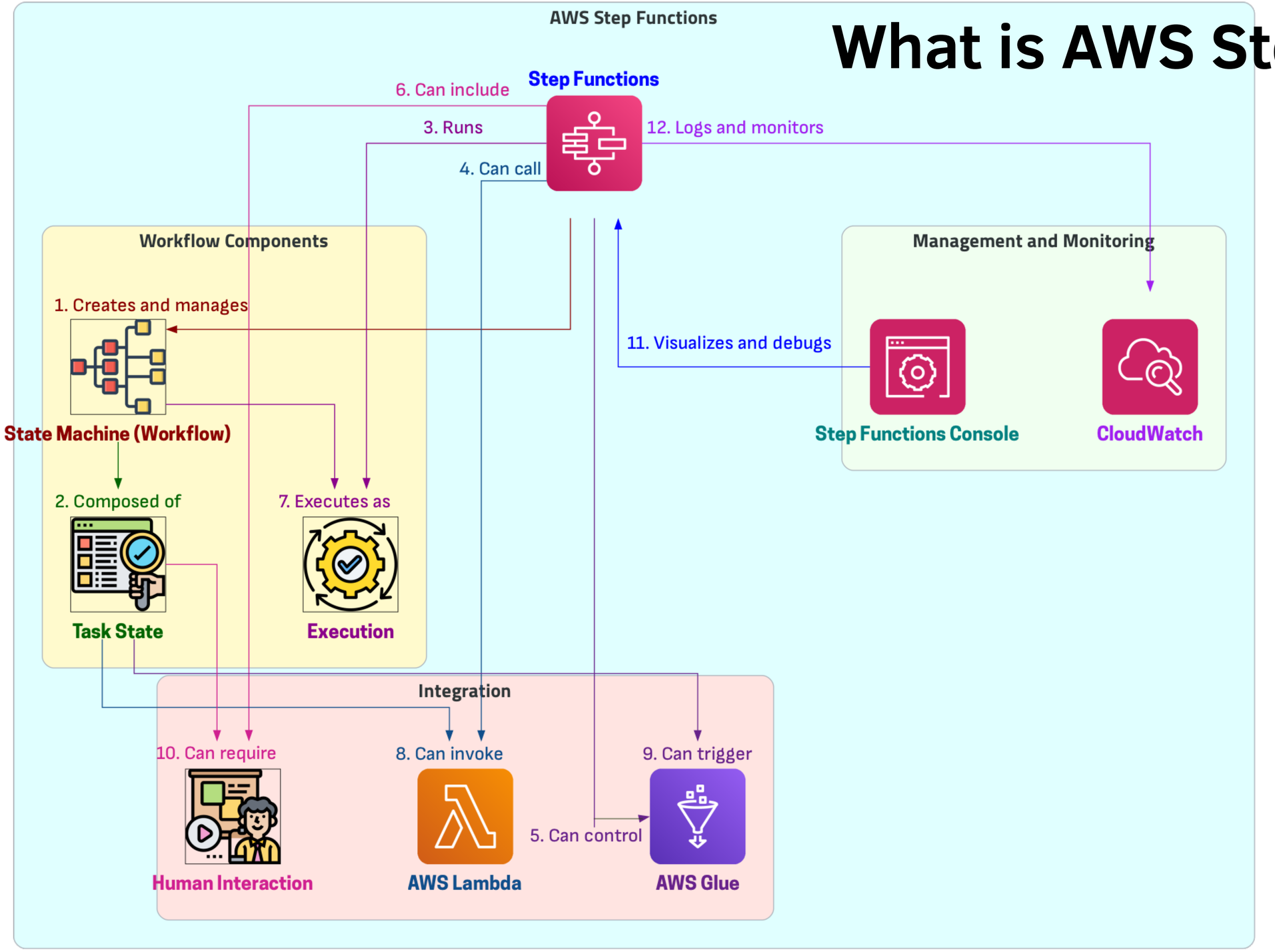
What is AWS Step Functions?



1️⃣. 🔄 Create workflows (state machines)
🏢 Distributed applications
⚙️ Automate processes
🎭 Orchestrate microservices
🔬 Data and ML pipelines

2️⃣. 🏗️ Components	
🔄 State machines (workflows)	🔀 Event-driven steps
🔷 States	🛠️ Task state: Unit of work
🏃 Executions: Running workflows	

What is AWS Step Functions?

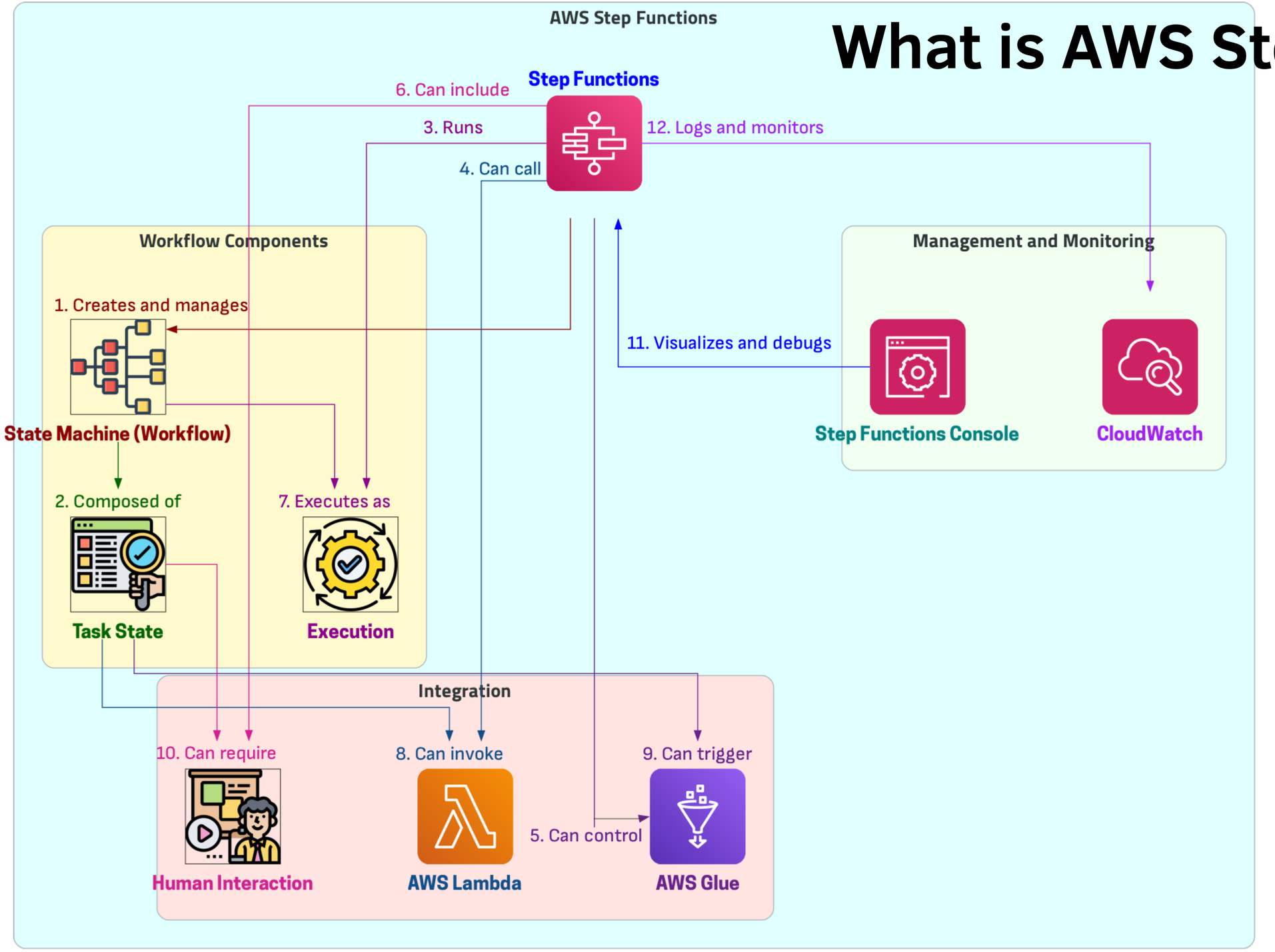


3️⃣. 👀 Console features
🖥️ Visualize workflows
✏️ Edit workflows
🐞 Debug workflows
🔍 Examine step states

4️⃣. 🔗 AWS service integration
⚡ Lambda: Perform tasks
🔧 Glue: ETL workflows

5️⃣. 🤖 Automation capabilities
⚙️ Complex processes
🎭 Microservices orchestration

What is AWS Step Functions?



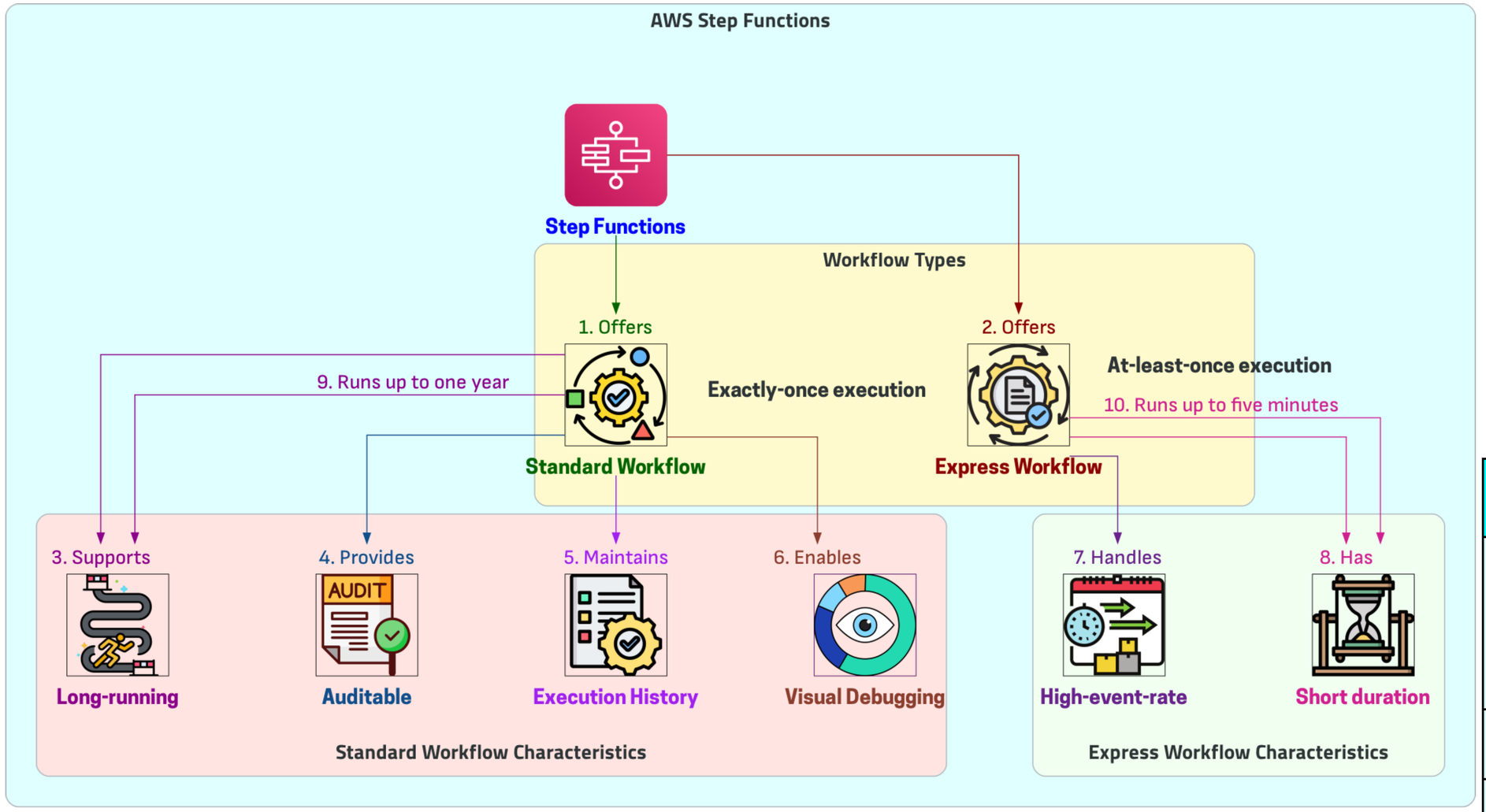
6️⃣. 📊 Data and ML pipelines

- 🔢 Data processing
- 🧠 Model training
- 🔮 Inference tasks

7️⃣. 👷 Human interaction support

- ✅ Manual approvals
- 📝 Human input stages

Standard and Express workflows types

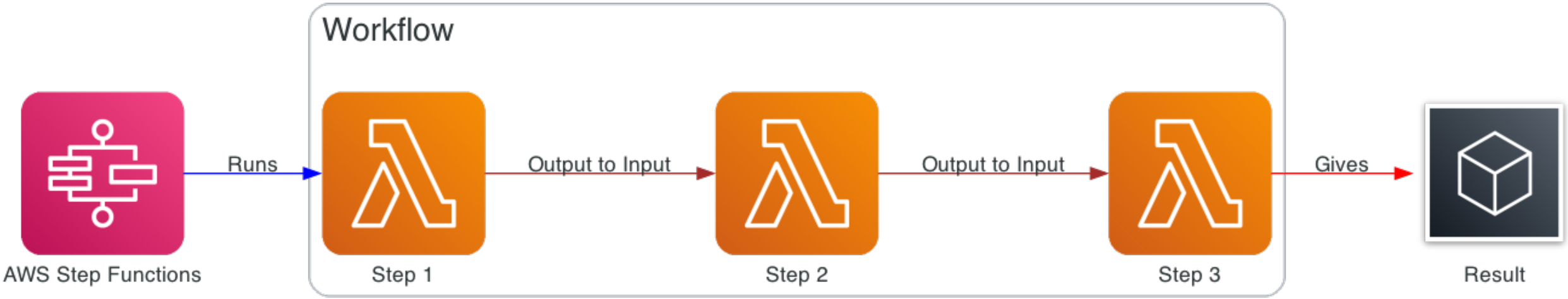


1. 🏛️ Standard workflows	
⏳ Long-running	
📜 Auditable	📊 Show execution history
	🦠 Visual debugging
🎯 Exactly-once execution	
🗓️ Run up to one year	

2. ⚡ Express workflows	
🚀 High-event-rate workloads	📊 Streaming data processing
	🌐 IoT data ingestion
🔁 At-least-once execution	
⏱️ Run up to five minutes	

Feature	Standard workflows	Express workflows
Execution rate	2,000 per second	100,000 per second
State transition rate	4,000 per second	Nearly unlimited
Pricing model	Priced by state transition	Priced by number and duration of executions
Execution history	Show execution history and visual debugging	Show execution history and visual debugging based on **log level**
History access	See execution history in Step Functions	Send execution history to CloudWatch
Service integrations	Support integrations with all services	Support integrations with all services
Integration patterns	Support *Request Response*, *Run a Job*, and *Wait for Callback* patterns (service-specific)	Support *Request Response* pattern for all services
Optimized integrations	Support integrations with all services	Support optimized integrations with some services

Use case #1: Function orchestration



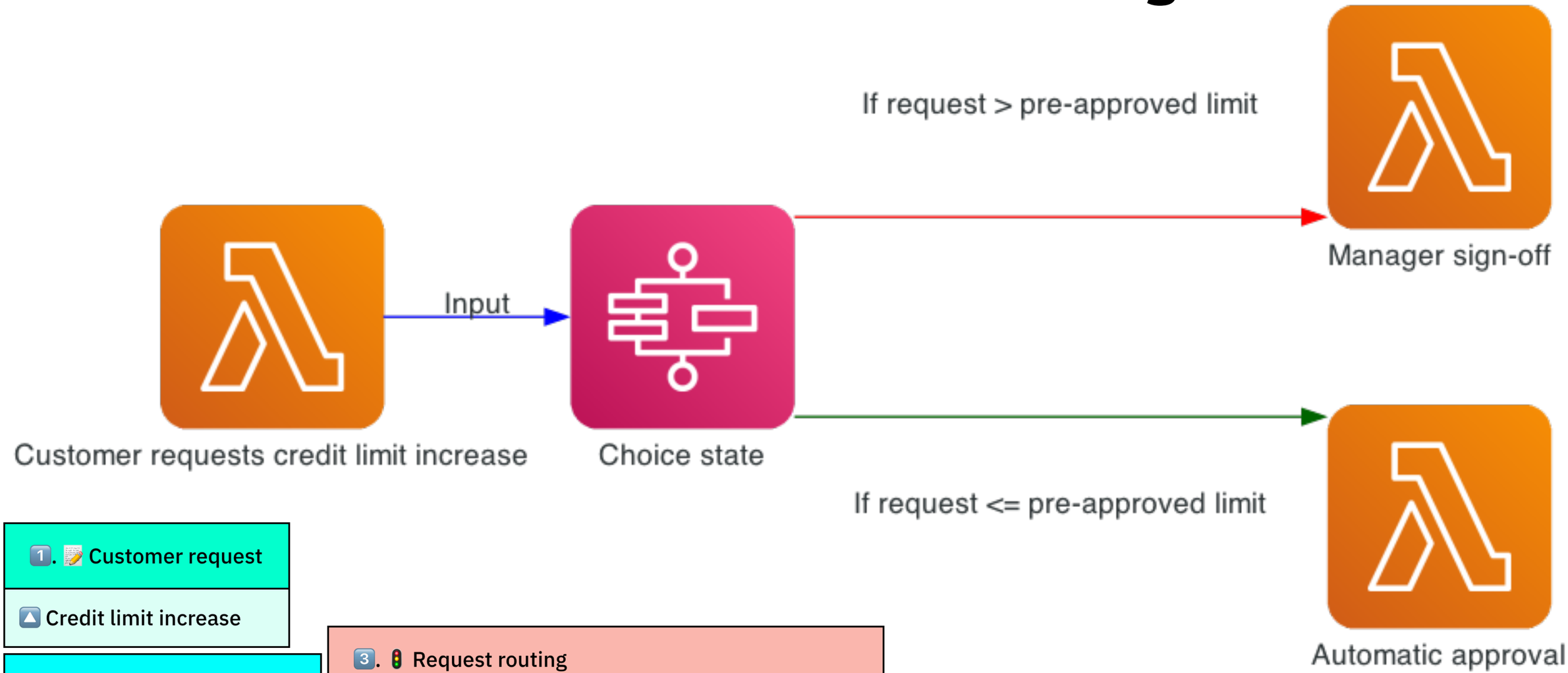
1️⃣. 🛠️ Create workflow
👥 Group of Lambda functions
🔢 Specific order

2️⃣. 🔗 Function interaction
📩 Output passes to next function
📩 Input receives from previous function

3️⃣. 🏁 Workflow completion
🔚 Last step
📊 Result generation

4️⃣. 👁️ Workflow visibility
🔍 See step interactions
✅ Verify intended function

Use case #2: Branching



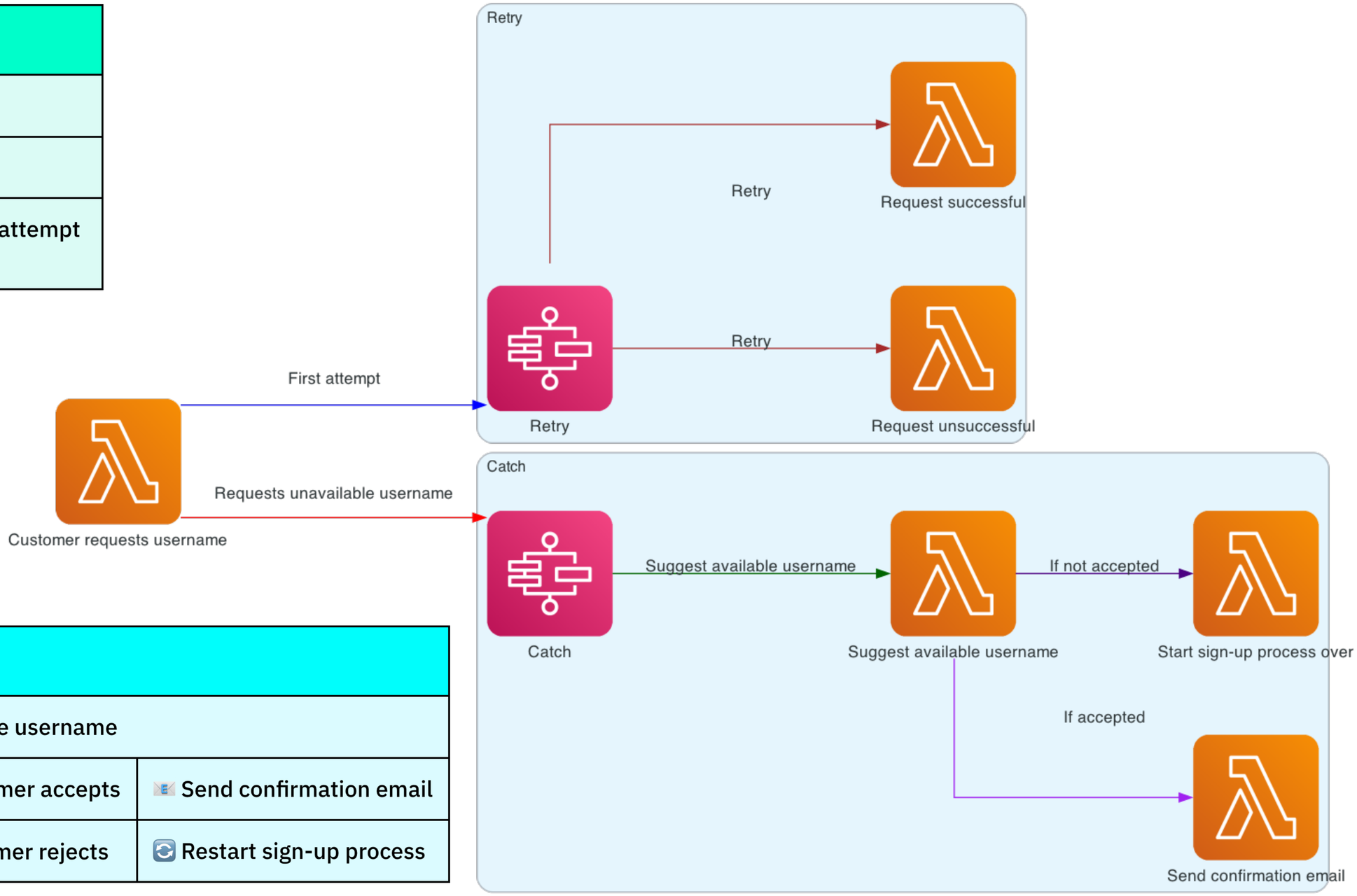
1️⃣. 📝 Customer request
🔼 Credit limit increase

2️⃣. 🔀 Choice state decision
📊 Based on input
💰 Compares to pre-approved limit

3️⃣. 🚦 Request routing	
📈 If above pre-approved limit	👨‍💼 Send to manager for sign-off
📉 If below pre-approved limit	🤖 Approve automatically

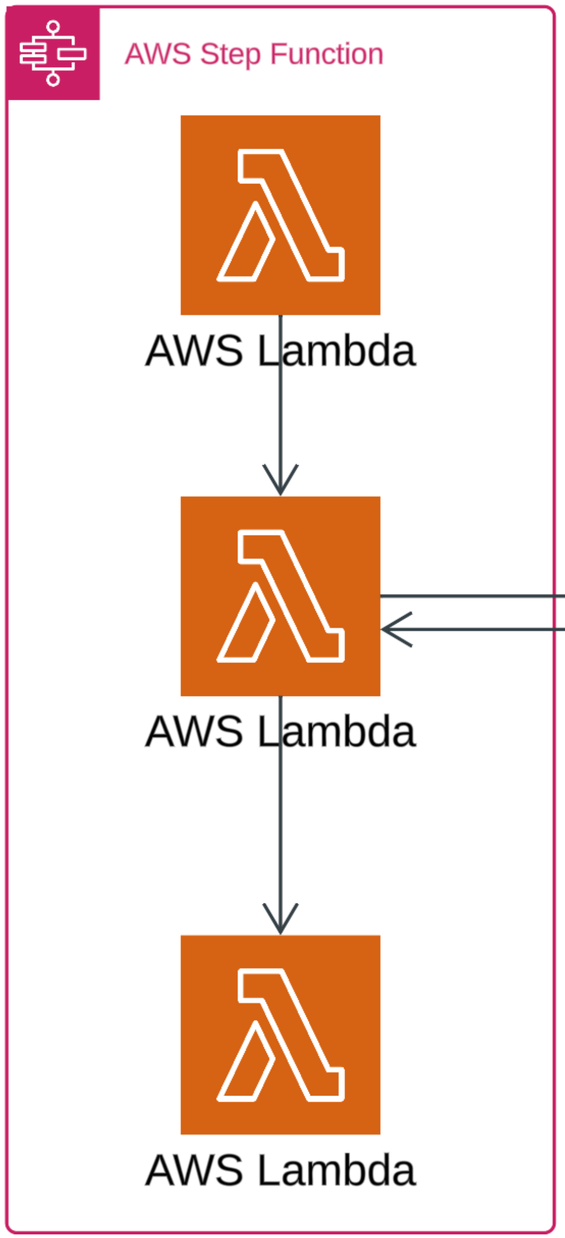
Use case #3: Error handling

1️⃣. 🔁 Retry	
🙋 Customer requests username	
✕ First attempt unsuccessful	
🔁 Retry statement	✅ Second attempt successful



2️⃣. 🎣 Catch		
🙋 Customer requests unavailable username		
💡 Suggest available username	✅ Customer accepts	📧 Send confirmation email
	✕ Customer rejects	🔁 Restart sign-up process

Use case #4: Human in the loop



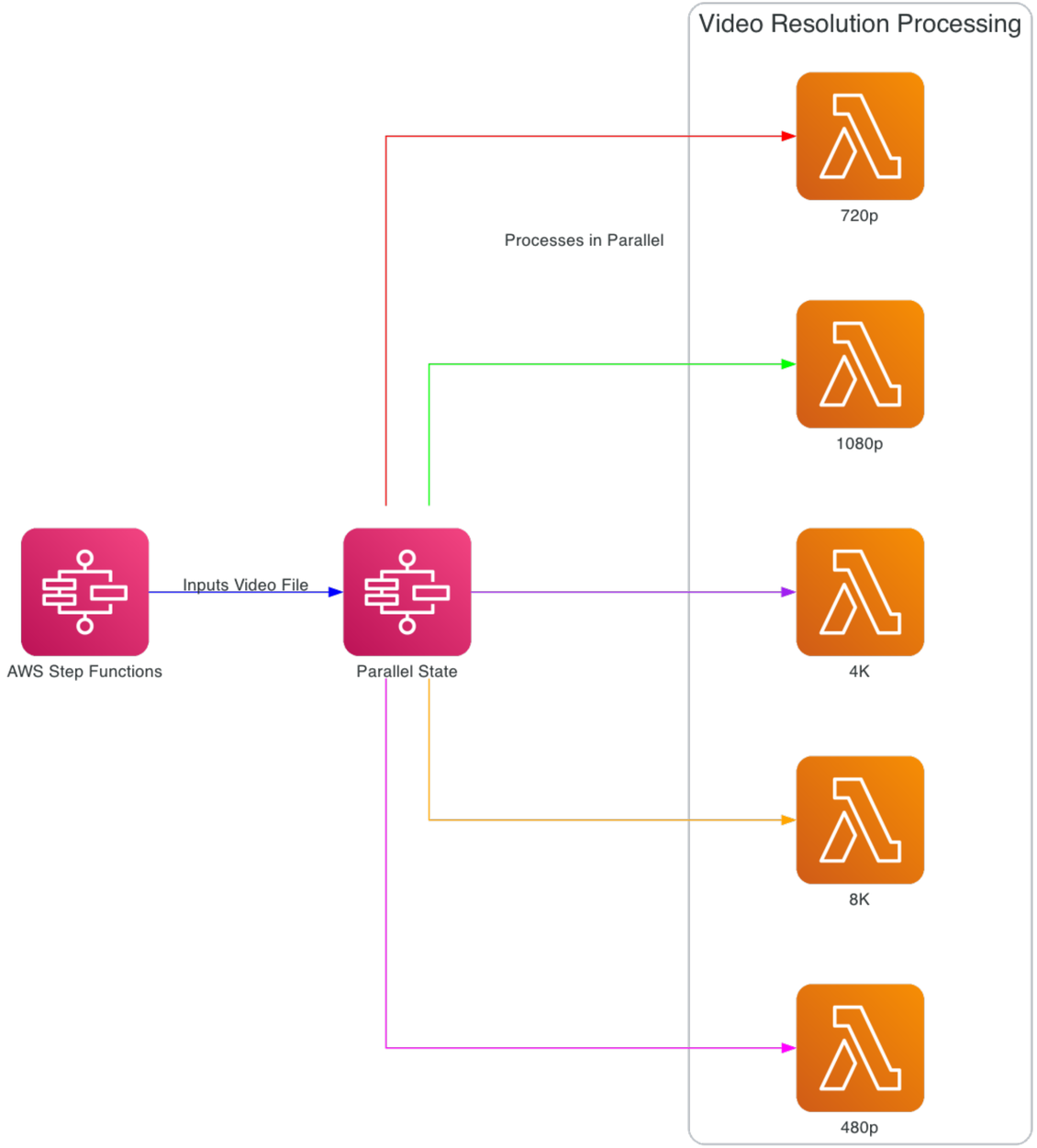
1️⃣. 📱 Customer initiates money transfer
👥 Sends money to friend
⏳ Waits for confirmation

2️⃣. 🔄 Step Functions and Lambda interaction
📞 Callback mechanism
🎫 Task token usage
💰 Lambda sends money

3️⃣. ✅ Transfer completion
🔔 Lambda reports back
🤝 Friend receives money

4️⃣. 📧 Confirmation process
▶️ Step Functions proceeds to next step
📩 Sends confirmation email to customer

Use case #5: Parallel processing



1️⃣. 🎞️ Customer input

🎥 Video file upload

2️⃣. 🖥️ Multiple display resolutions

5️⃣ Five different resolutions

📱 Support for multiple devices

3️⃣. ⏸️ Parallel state in Step Functions

🔀 Simultaneous processing

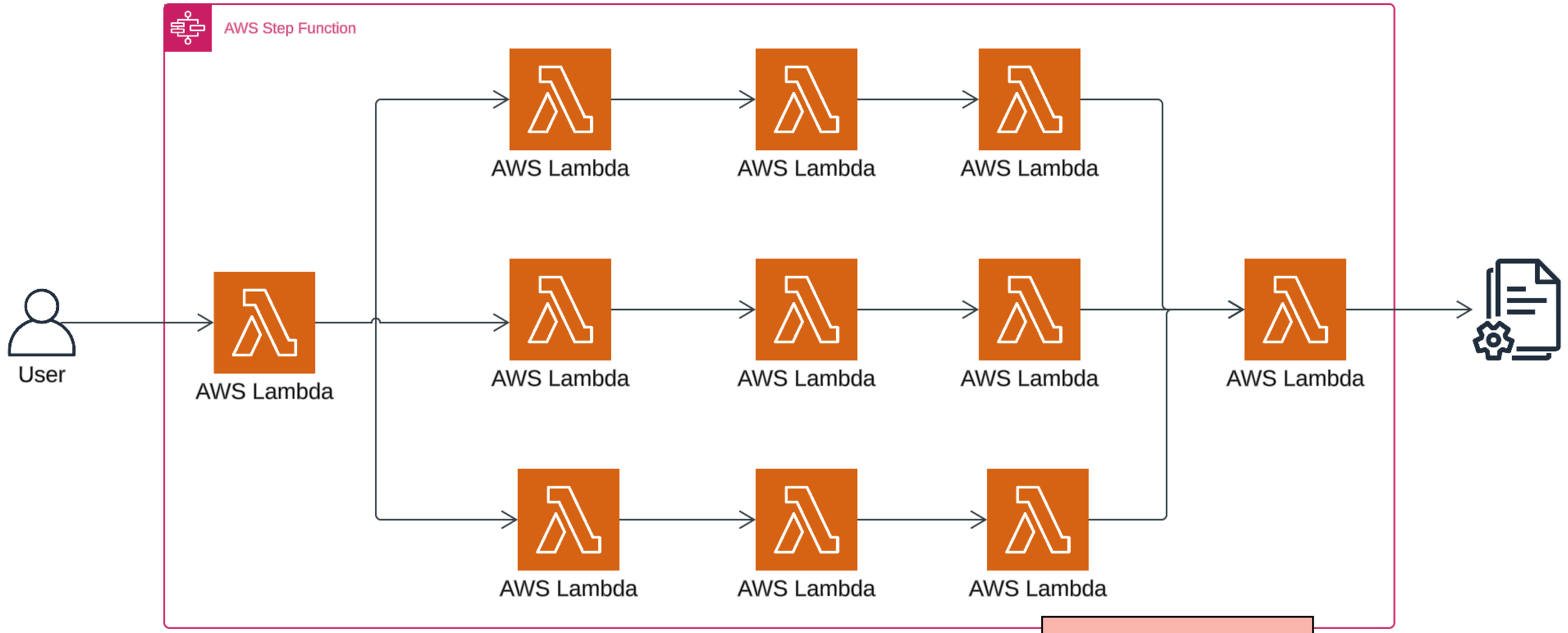
⚡ Efficient conversion

4️⃣. 🔧 Lambda processing

🛠️ Converts video file

🔁 Generates multiple resolutions

Use case #6: Dynamic parallelism



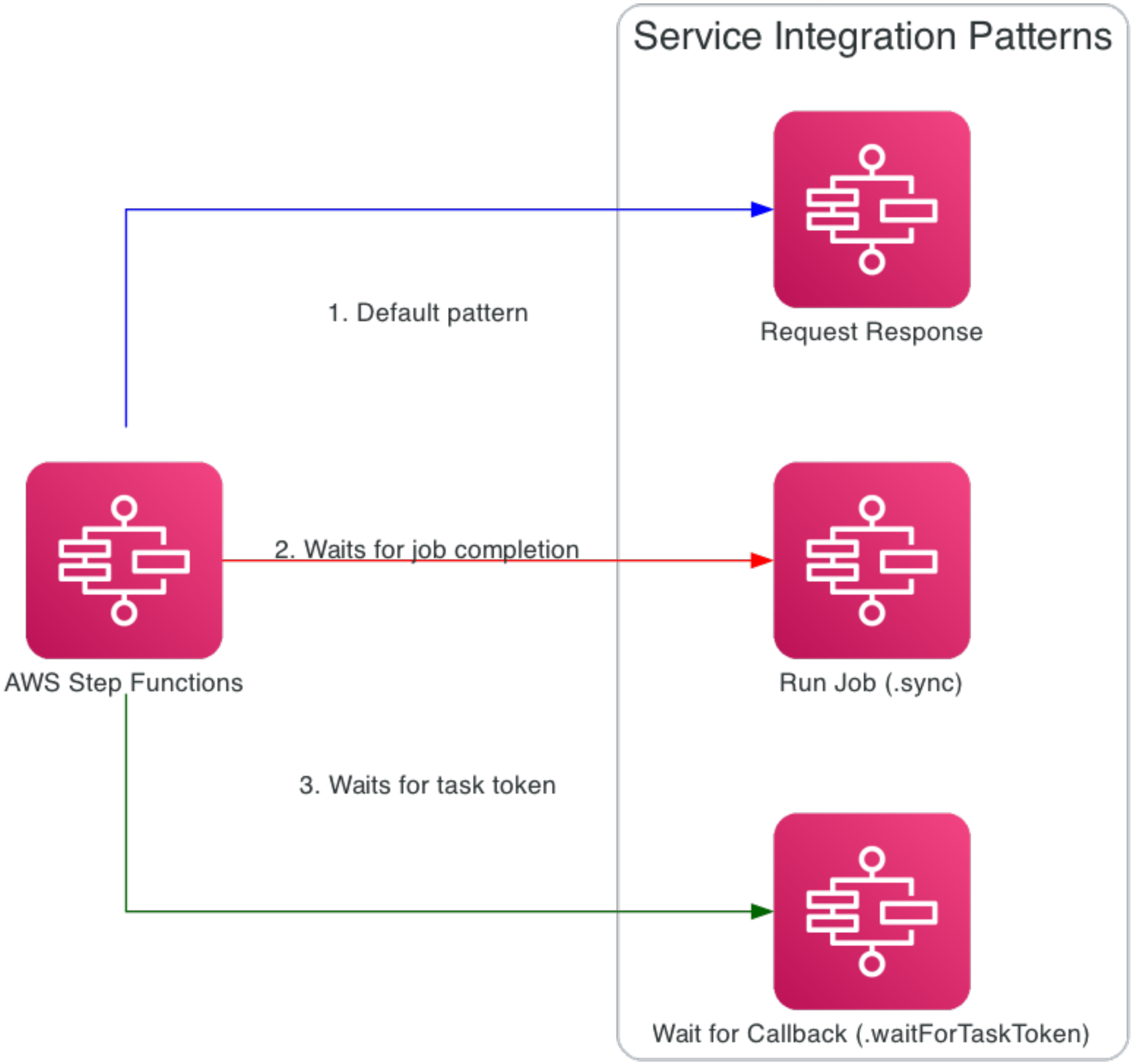
1️⃣. 📦 Customer order
3 Three items ordered

2️⃣. 🗺️ Map state in Step Functions
⚡ Parallel processing
🔄 Iterates over each item

3️⃣. 🔍 Item processing
✅ Check availability
🧺 Gather item
📦 Package for delivery

4️⃣. 📧 Order confirmation
✉️ Send email to customer
🔢 Include tracking information

Service integrations Patterns



1️⃣. 🤝 Multiple AWS service integrations

🔌 Combines Step Functions with other services

2️⃣. 📞 Request a response (default)

🚀 Calls service

✅ Progresses after HTTP response

3️⃣. 🏃 Run a job (.sync)

📞 Calls service

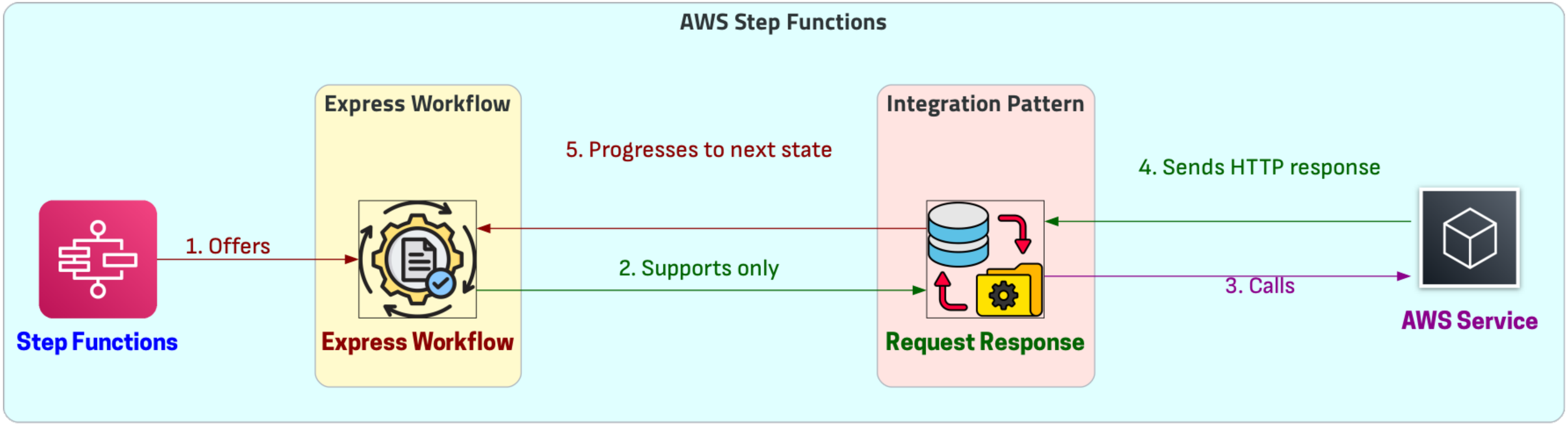
⏳ Waits for job completion

4️⃣. 🪪 Wait for callback (task token)

🔑 Calls service with task token

⏰ Waits for token return

Service integrations Patterns



1️⃣. 📞 Request a response pattern
🚀 Step Functions calls service
📩 Receives HTTP response
➡️ Progresses to next state

Optimized integrations

Thanks for Watching