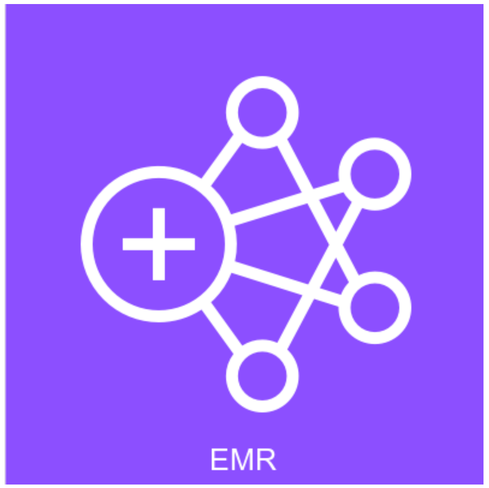
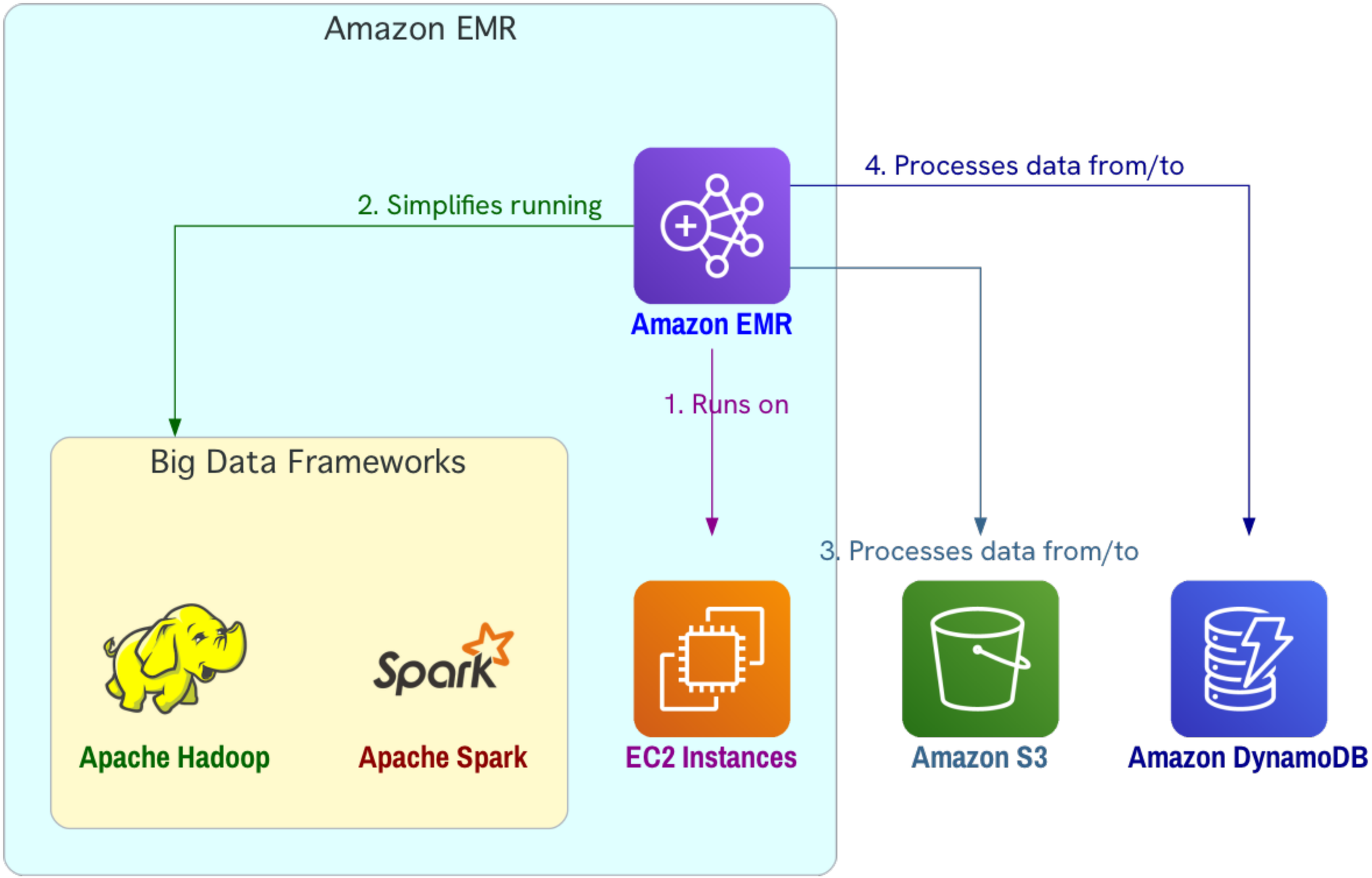


Amazon EMR

Table of Contents

What is Amazon EMR?



1️⃣. 🖥️ Managed cluster platform

📊 Simplifies big data frameworks on AWS

🔄 Formerly Amazon Elastic MapReduce

2️⃣. 🦣 Supports big data frameworks

💻 Apache Hadoop

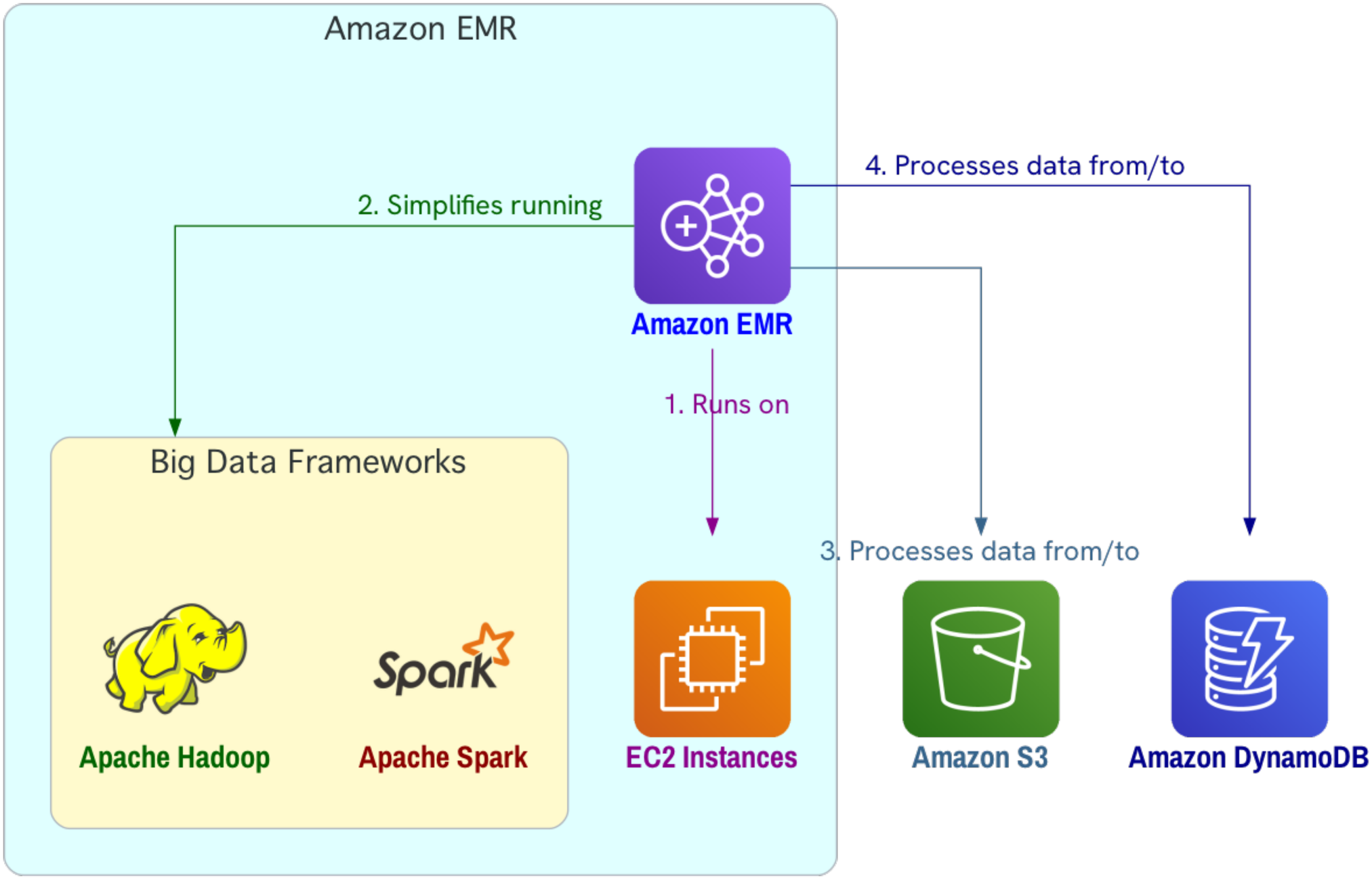
⚡ Apache Spark

3️⃣. 📊 Process and analyze vast data

🔬 Efficient large-scale data processing

📈 Data analysis capabilities

What is Amazon EMR?



4️⃣. 📈 Analytics and business intelligence

📊 Data analytics purposes

🧠 Powers BI applications

5️⃣. 🔄 Data transformation and movement

🔀 Transform large data volumes

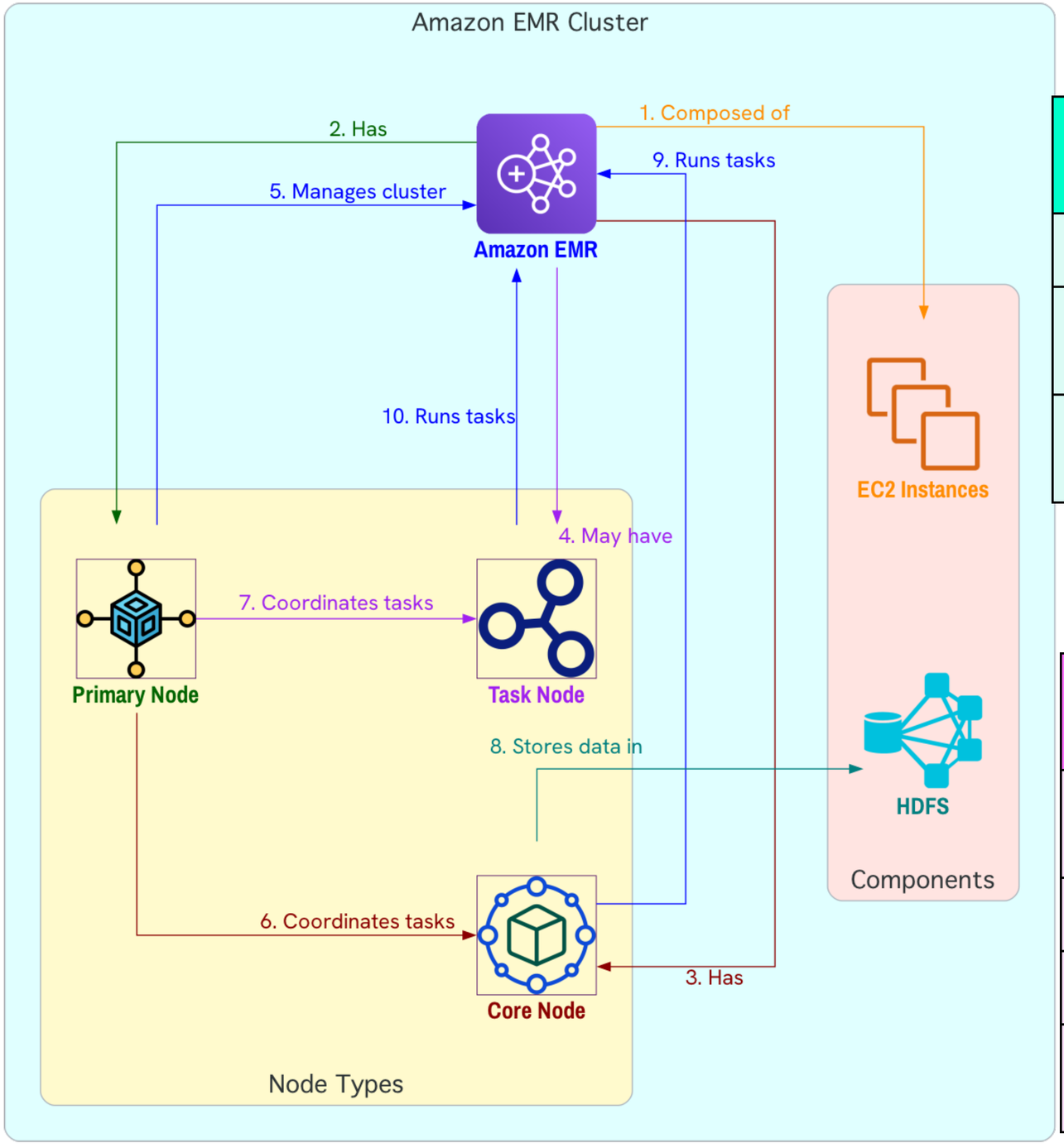
➡️ Move data between systems

6️⃣. 🔗 AWS service integration

🪣 Amazon S3 integration

⬛ Amazon DynamoDB integration

Understanding clusters and nodes



1️⃣. 🖥️ Cluster: Collection of EC2 instances
🏢 Group of EC2 instances
🔧 Central component of EMR
📊 Enables distributed big data processing

2️⃣. 📊 Node: Individual instance in a cluster
🧱 Building blocks of cluster
🎭 Specific role in distributed application

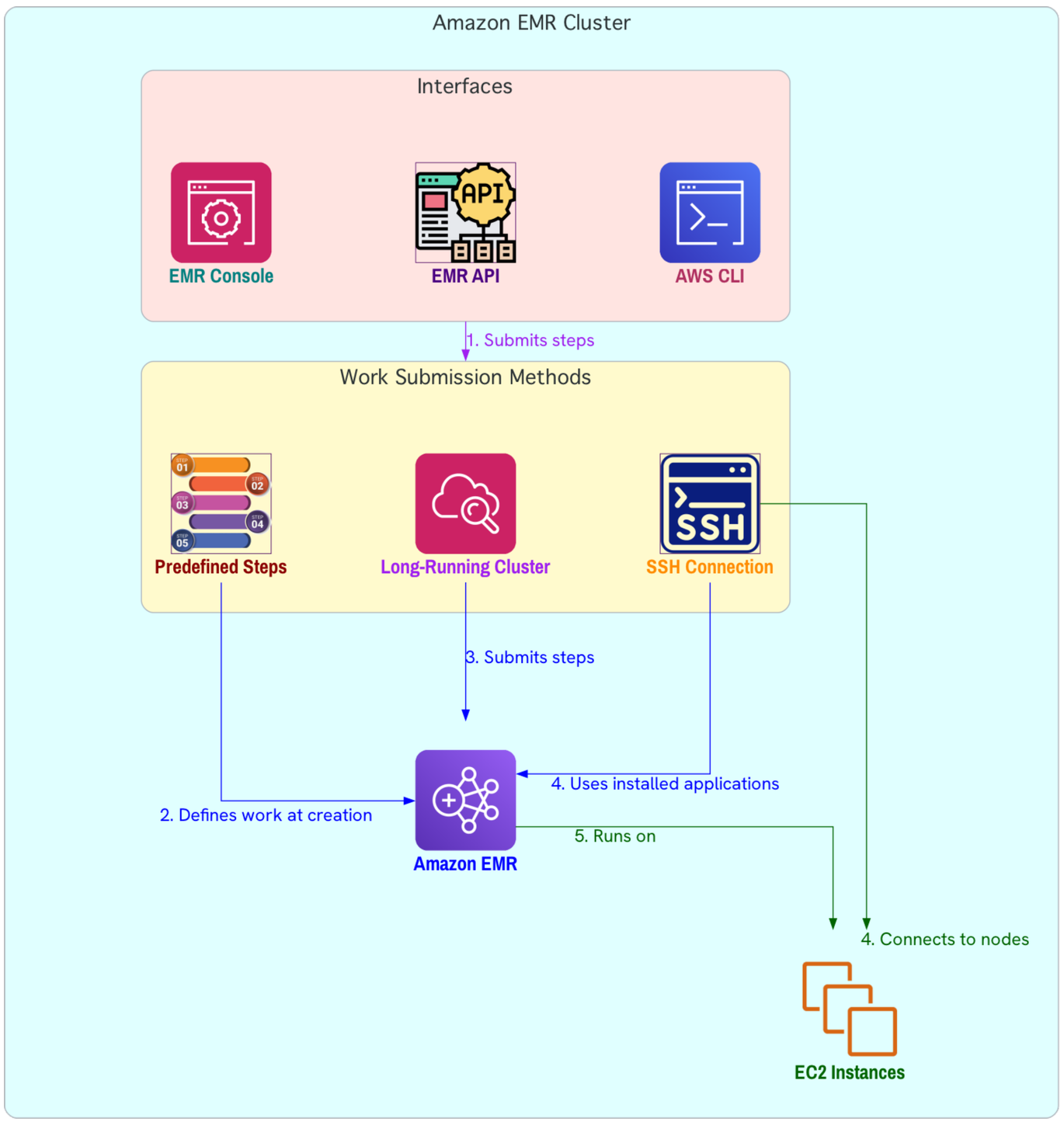
3️⃣. 🎭 Node types: Different roles in cluster
👥 Assigned roles within cluster
💻 Specific software components installed
🐘 Example: Apache Hadoop

4️⃣. 🧠 Primary node: Manages cluster
🎛️ Coordinates data and task distribution
📊 Tracks task status
🩺 Monitors cluster health
1️⃣ One per cluster (minimum)

5️⃣. 💪 Core node: Runs tasks & stores data
🏃 Runs tasks
💾 Stores data in HDFS
🔢 At least one in multi-node clusters

6️⃣. 🚀 Task node: Runs tasks only (optional)
🏋️ Provides additional compute capacity
🚫 Does not store data in HDFS
⚖️ Increases processing without adding storage

Submitting work to a cluster

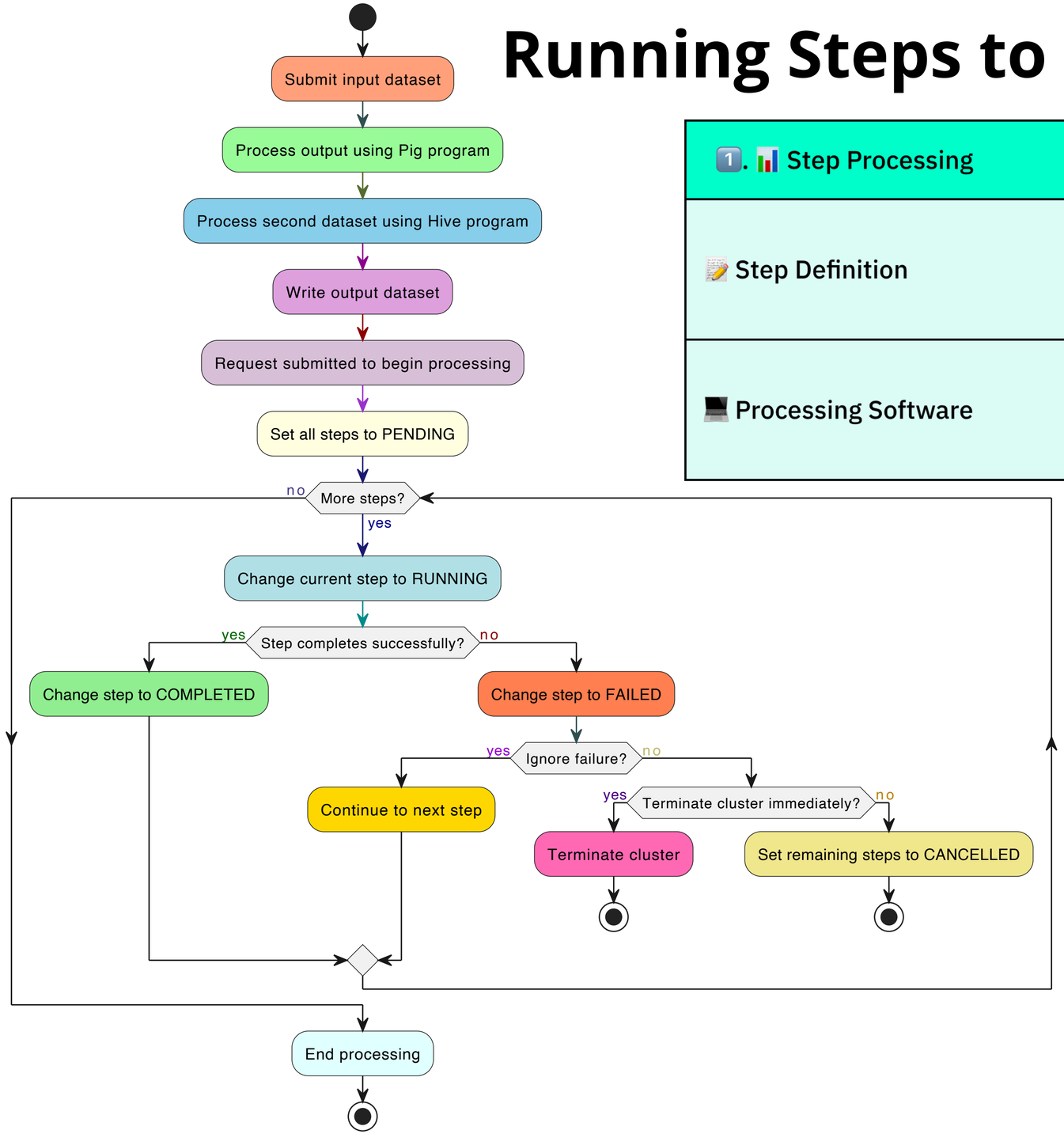


1️⃣. 📋 Define work as steps during cluster creation
📝 Specify functions as steps
🔢 Process set amount of data
🛑 Terminate after completion
📦 Ideal for batch processing

2️⃣. 🔄 Submit steps to long-running clusters	
🛠️ Use various tools	🖥️ Amazon EMR console
	🔌 Amazon EMR API
	⌨️ AWS CLI
📊 Steps contain one or more jobs	
🔁 Flexible for ongoing workloads	

3️⃣. 🖥️ Connect via SSH and use application interfaces
🔒 SSH to primary and other nodes
🖥️ Use installed application interfaces
📜 Submit tasks via scripts
🔍 Interactive work possible
🛠️ Ideal for complex tasks and debugging

Running Steps to Process Data in Amazon EMR

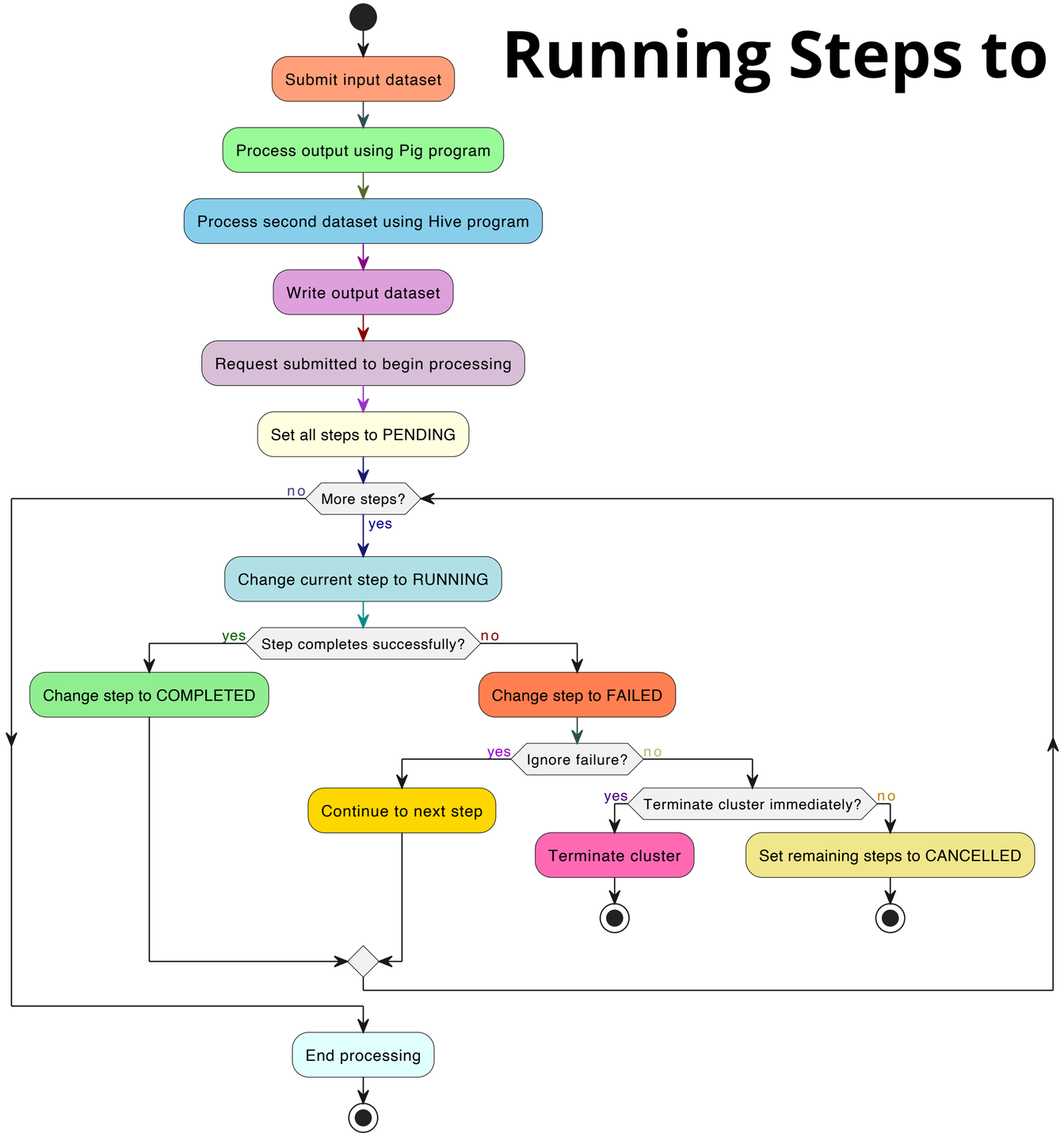


1. 📊 Step Processing	
📝 Step Definition	🔧 Unit of work
	📊 Data manipulation instructions
💻 Processing Software	🖥️ Installed on cluster
	⚙️ Executes instructions

2. 🔢 Example Process
1️⃣ Submit input dataset
2️⃣ Process with Pig program
3️⃣ Process second dataset with Hive
4️⃣ Write output dataset

3. 🔁 Data Flow	
💾 Input Data Storage	☁️ Amazon S3
	🟫 HDFS
🔄 Data Processing	➡️ Step-by-step sequence
📤 Output Data Storage	☁️ Amazon S3 bucket

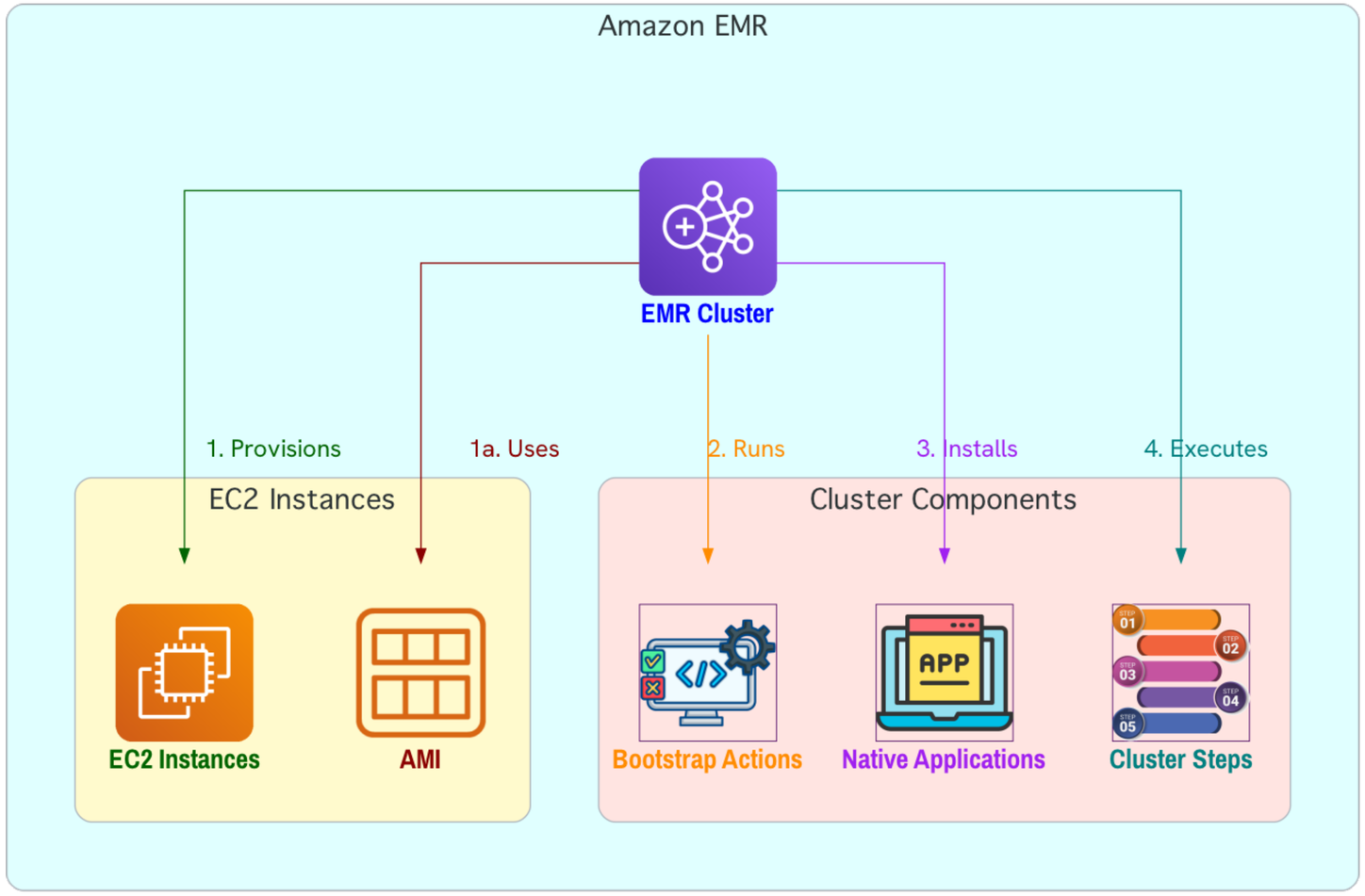
Running Steps to Process Data in Amazon EMR



4️⃣. 🔄 Step Execution Sequence	
🏁 Initiation	📩 Submit processing request
⏳ Initial State	🟡 All steps PENDING
▶️ First Step Execution	🟢 First step RUNNING
	🟡 Other steps PENDING
✅ Step Completion	🔵 Completed step COMPLETED
	▶️ Next step starts RUNNING
🔁 Process Repetition	🔄 Repeat for all steps
🏁 Processing End	✅ All steps complete

5️⃣. ✕ Step Failure Handling	
🔴 Failure State	✕ Failed step FAILED
🚫 Default Behavior	🛑 Remaining steps CANCELLED
🔀 Alternative Options	➡️ Ignore failure and proceed
	⛔ Terminate cluster immediately

Amazon EMR Cluster Lifecycle

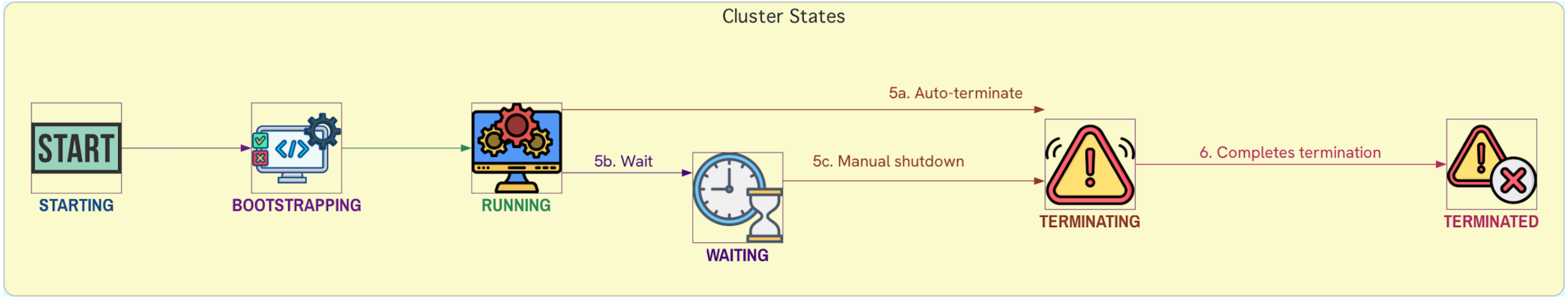


1️⃣. 🚀 STARTING: Provision EC2 instances
🖥️ Provisions EC2 instances
💿 Uses default or custom AMI

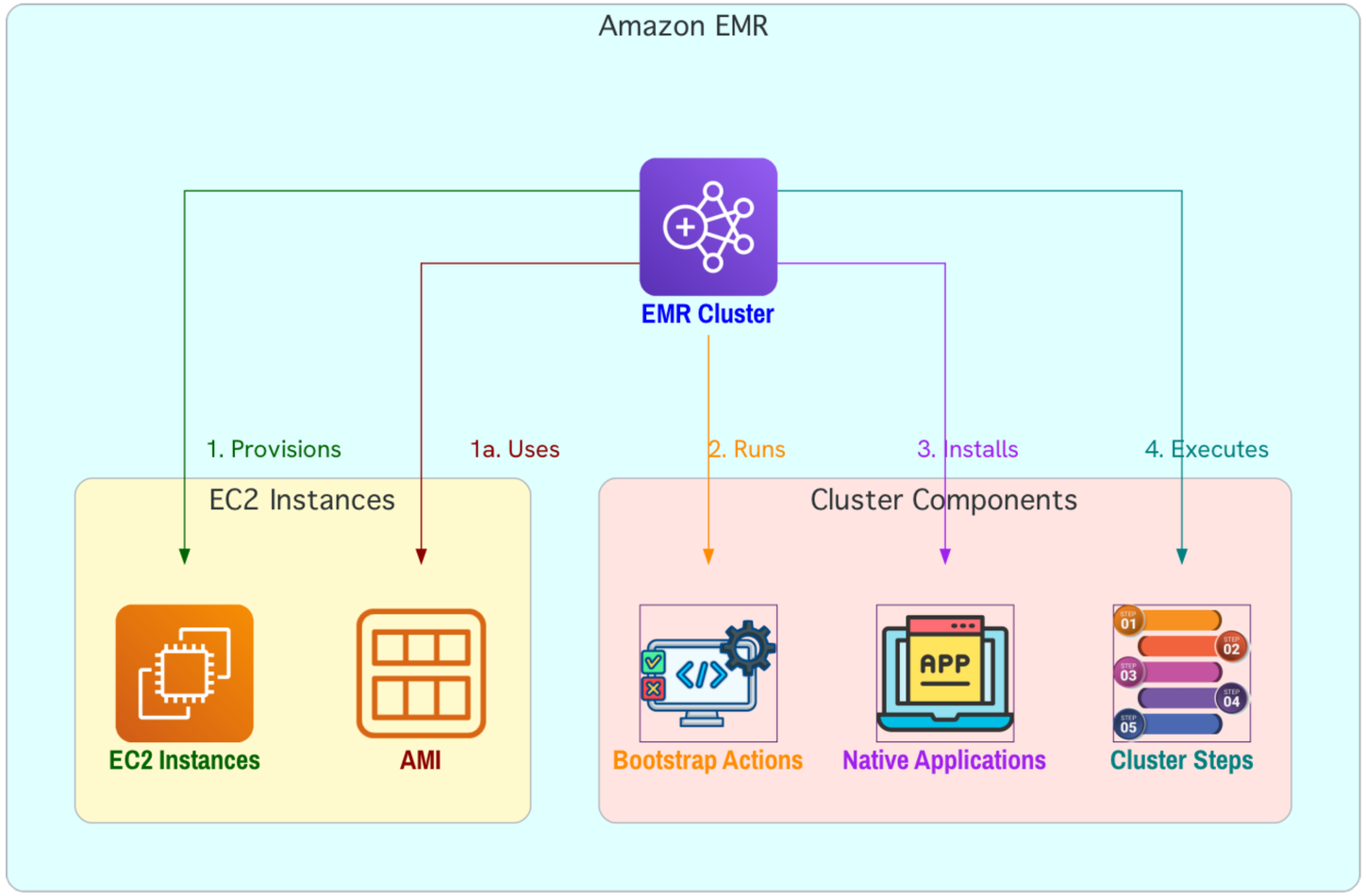
2️⃣. 🛠️ BOOTSTRAPPING: Run bootstrap actions
🔧 Executes specified actions
📥 Installs custom applications
🔨 Performs customizations

3️⃣. 📦 Install native applications
🐘 Hadoop
🐝 Hive
⚡ Spark

4️⃣. 🏃 RUNNING: Execute cluster steps
🔗 Connect to cluster instances
▶️ Run specified steps
➕ Submit additional steps



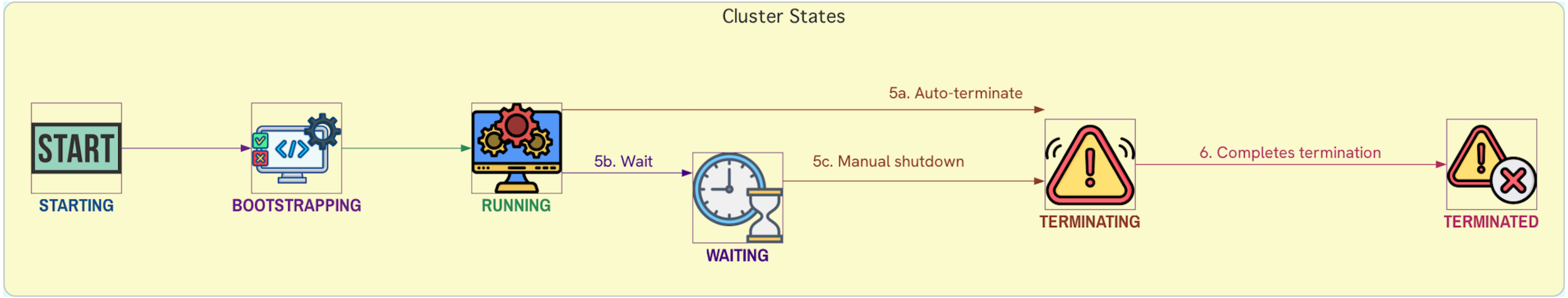
Amazon EMR Cluster Lifecycle

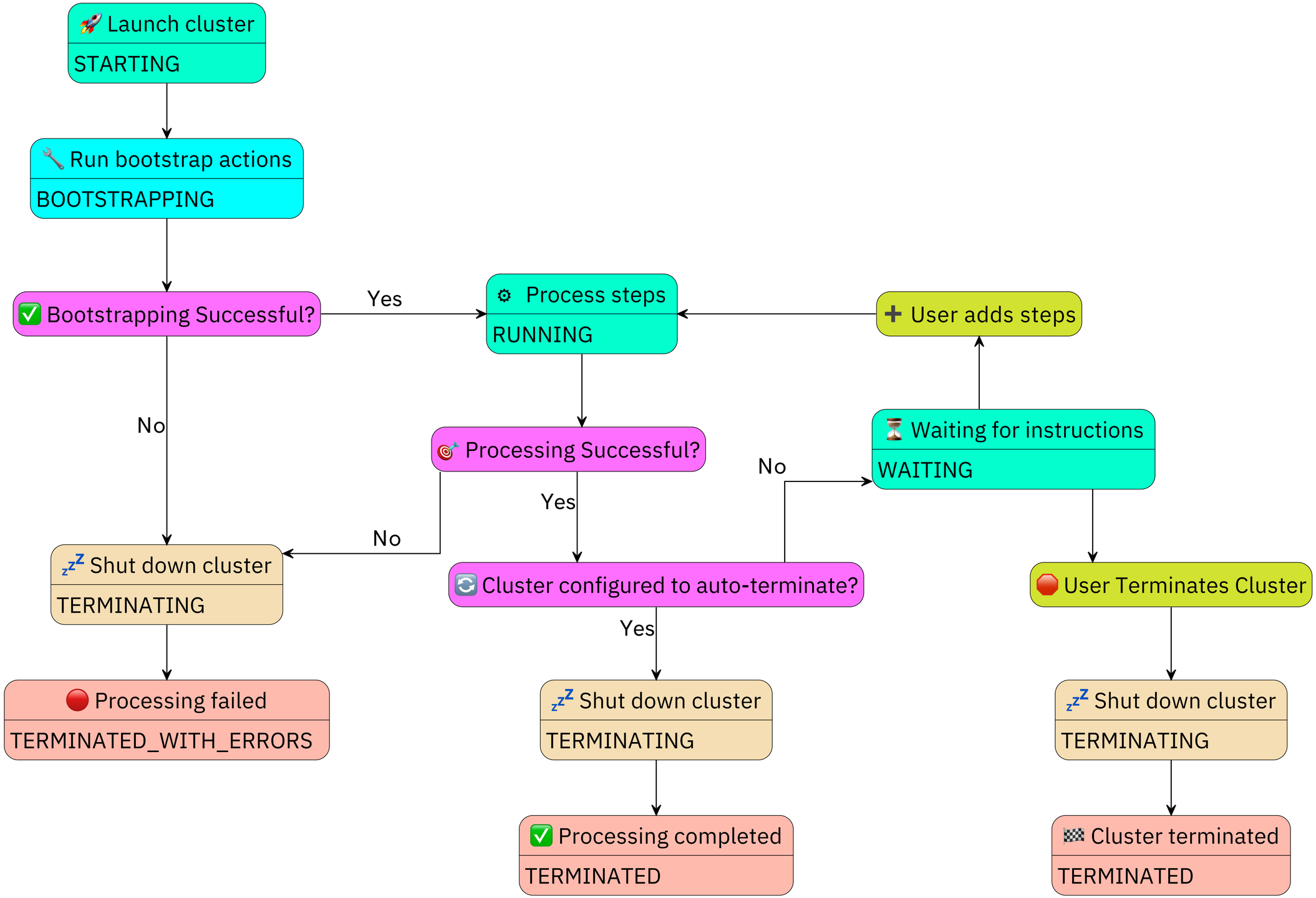


5️⃣. ⏳ WAITING: Cluster awaits next action
💤 Idle state
➡️ SOON Ready for further instructions

6️⃣. 🛑 TERMINATING: Cluster shuts down
🔄 Auto-terminate after last step
🖐️ Manual shutdown initiation

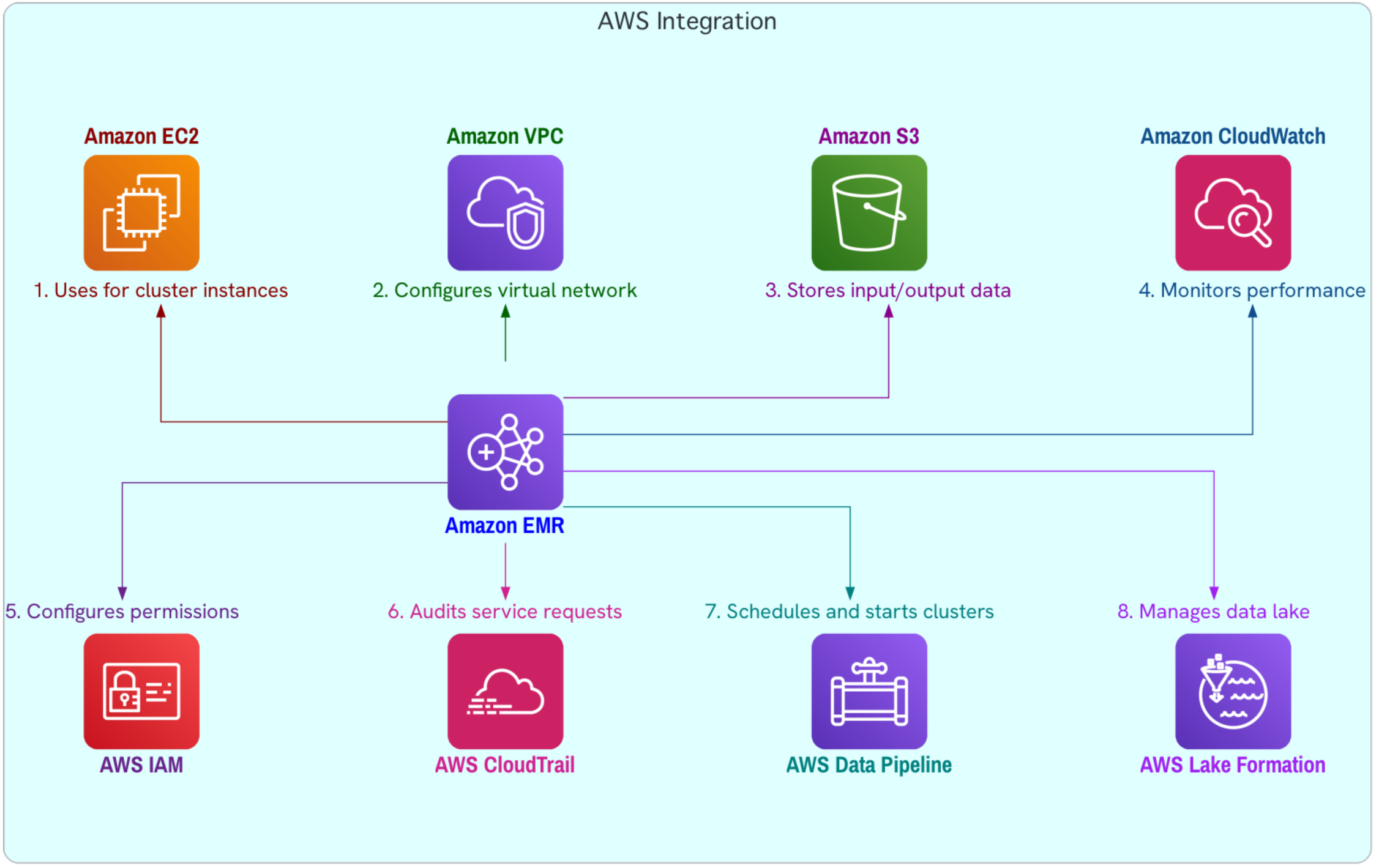
7️⃣. ⚫ TERMINATED: Cluster process complete
🔒 Resources released
🏁 Cluster no longer active



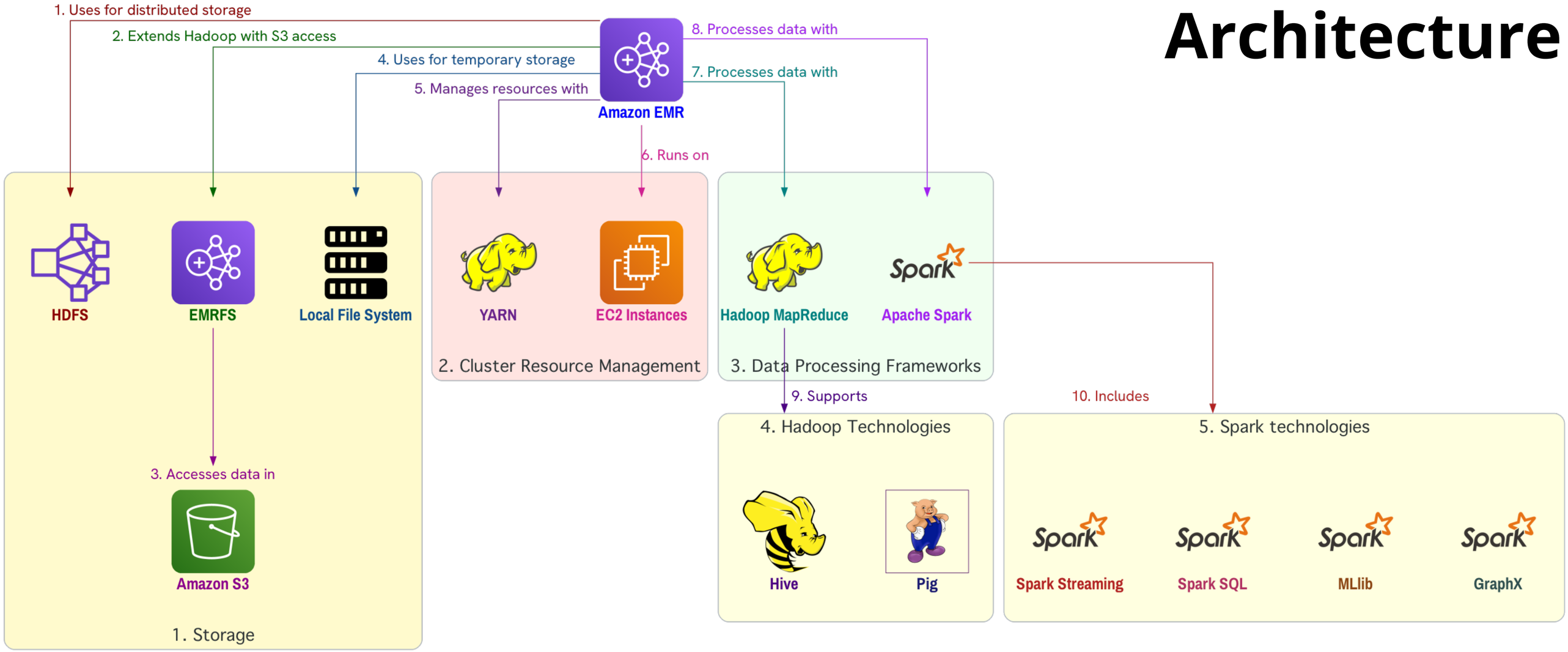

🚀 Launch cluster
STARTING
🔧 Run bootstrap actions
BOOTSTRAPPING
✅ Bootstrapping Successful?
Yes
⚙ Process steps
RUNNING
➕ User adds steps
No
🎯 Processing Successful?
⏳ Waiting for instructions
WAITING
No
Yes
No
💤 Shut down cluster
TERMINATING
🔄 Cluster configured to auto-terminate?
🛑 User Terminates Cluster
Yes
🔴 Processing failed
TERMINATED_WITH_ERRORS
💤 Shut down cluster
TERMINATING
💤 Shut down cluster
TERMINATING
✅ Processing completed
TERMINATED
🏁 Cluster terminated
TERMINATED

Amazon EMR Cluster Failure Handling

🚨 Amazon EMR Cluster Failure Handling	
1️⃣. 💥 Cluster Failure Consequences	🔚 Cluster termination
	💻 Instance termination
2️⃣. 🛡️ Termination Protection	🚫 Prevents automatic termination
	💾 Allows data retrieval
3️⃣. 📊 Cluster State	✕ TERMINATED_WITH_ERRORS
4️⃣. 🔄 Recovery Process	📩 Retrieve data
	🔓 Remove termination protection
	🛑 Manually terminate cluster

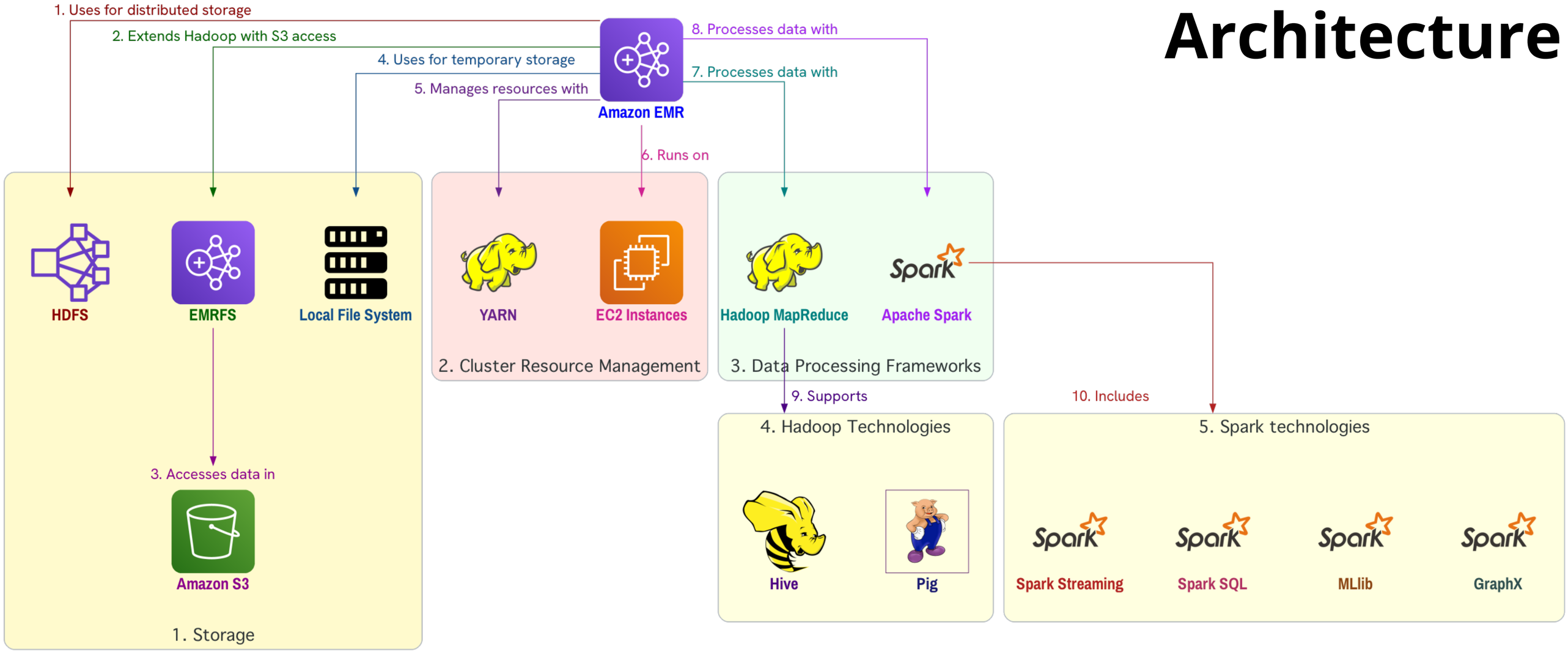

AWS Integration
Amazon EC2
Amazon VPC
Amazon S3
Amazon CloudWatch
1. Uses for cluster instances
2. Configures virtual network
3. Stores input/output data
4. Monitors performance
Amazon EMR
5. Configures permissions
6. Audits service requests
7. Schedules and starts clusters
8. Manages data lake
AWS IAM
AWS CloudTrail
AWS Data Pipeline
AWS Lake Formation

Architecture



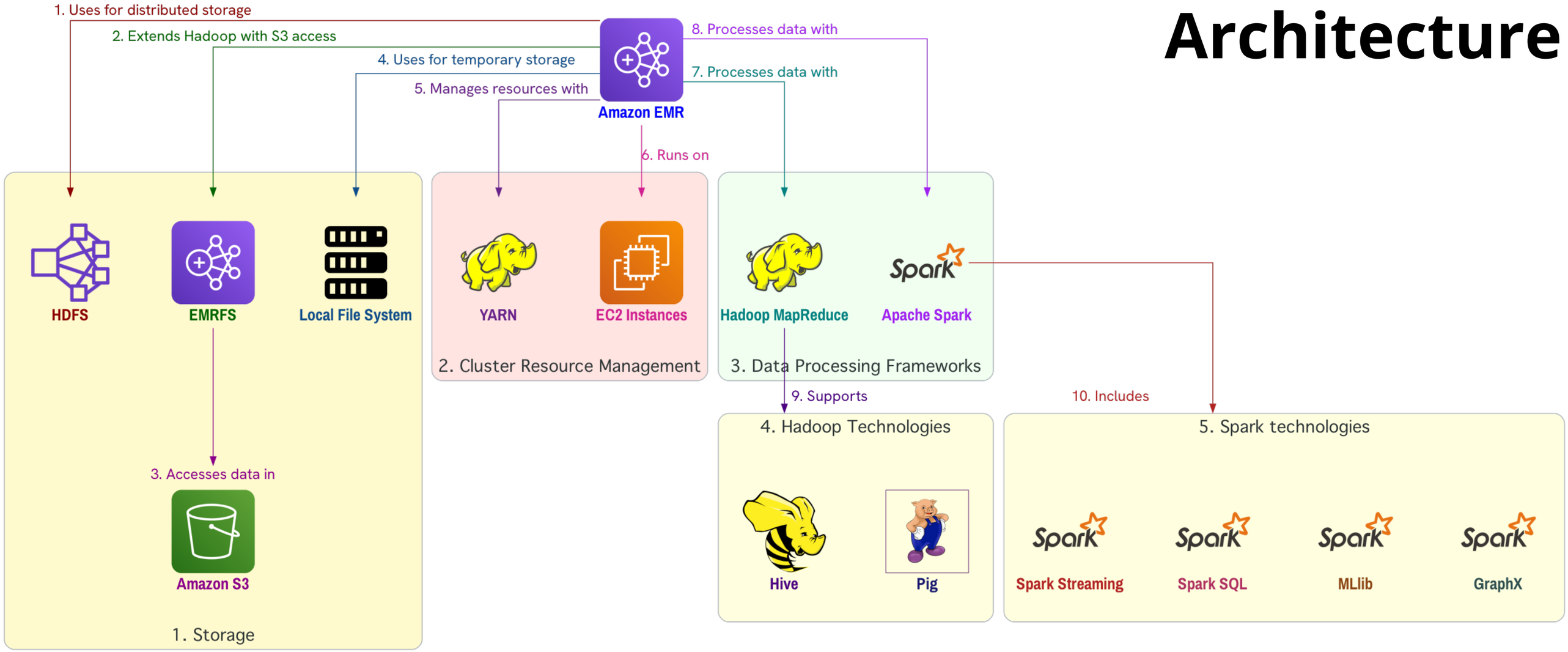
1️⃣ . 💾 **Storage Layer**: 📁 HDFS: 🔄 Distributed, scalable, 🛡️ Fault tolerance, 🧮 Intermediate results; 🌐 EMRFS: 🔗 Direct S3 access, 📥📤 Input/output data; 💽 Local File System: ☁️ EC2 instance store, ⏳ Temporary storage

Architecture



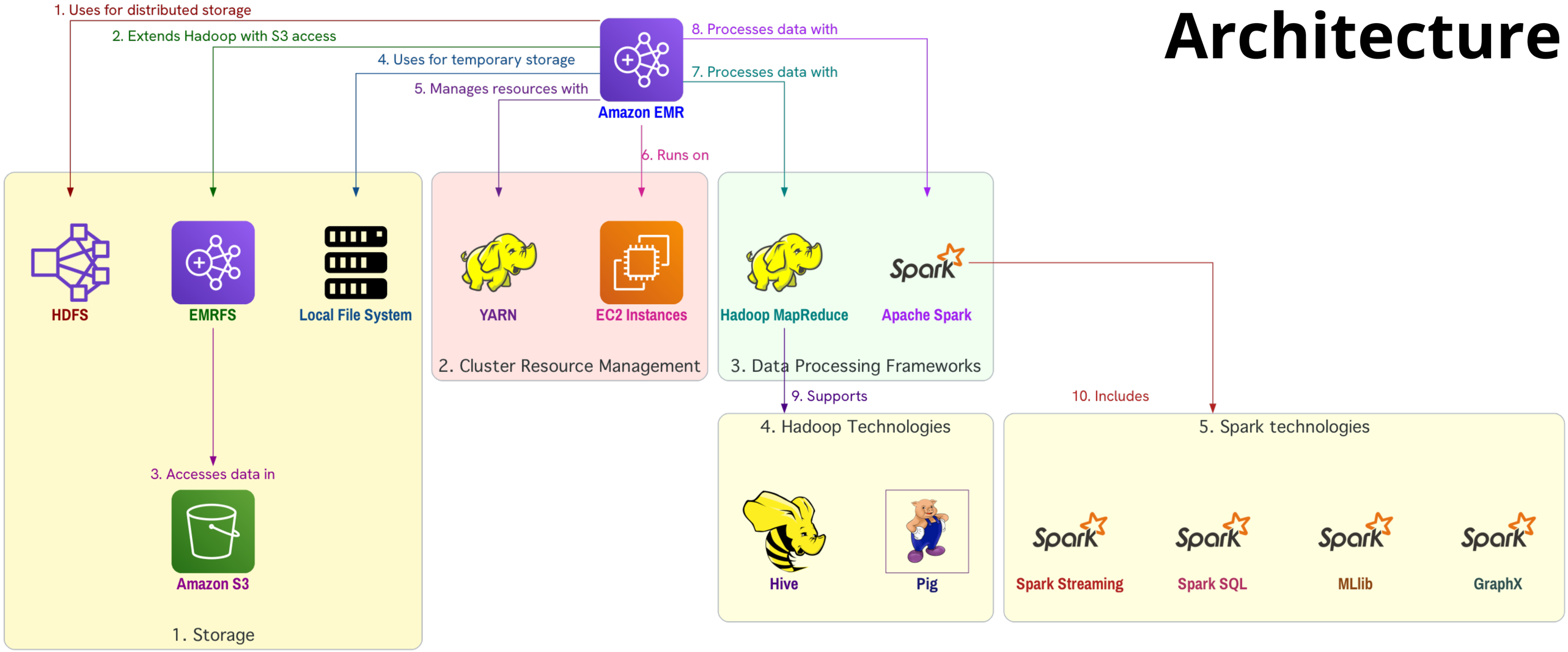
2️⃣ . 🧠 Resource Management: 🧶 YARN: 🎛️ Central resource management, 🔀 Multiple frameworks; 🤖 EMR Agent: 🔧 Administers YARN, 🏥 Maintains cluster health, 📡 Communicates with EMR; 🎯 Spot Instance optimization: 🏠 Core nodes for app masters, 🔄 Job continuity

Architecture



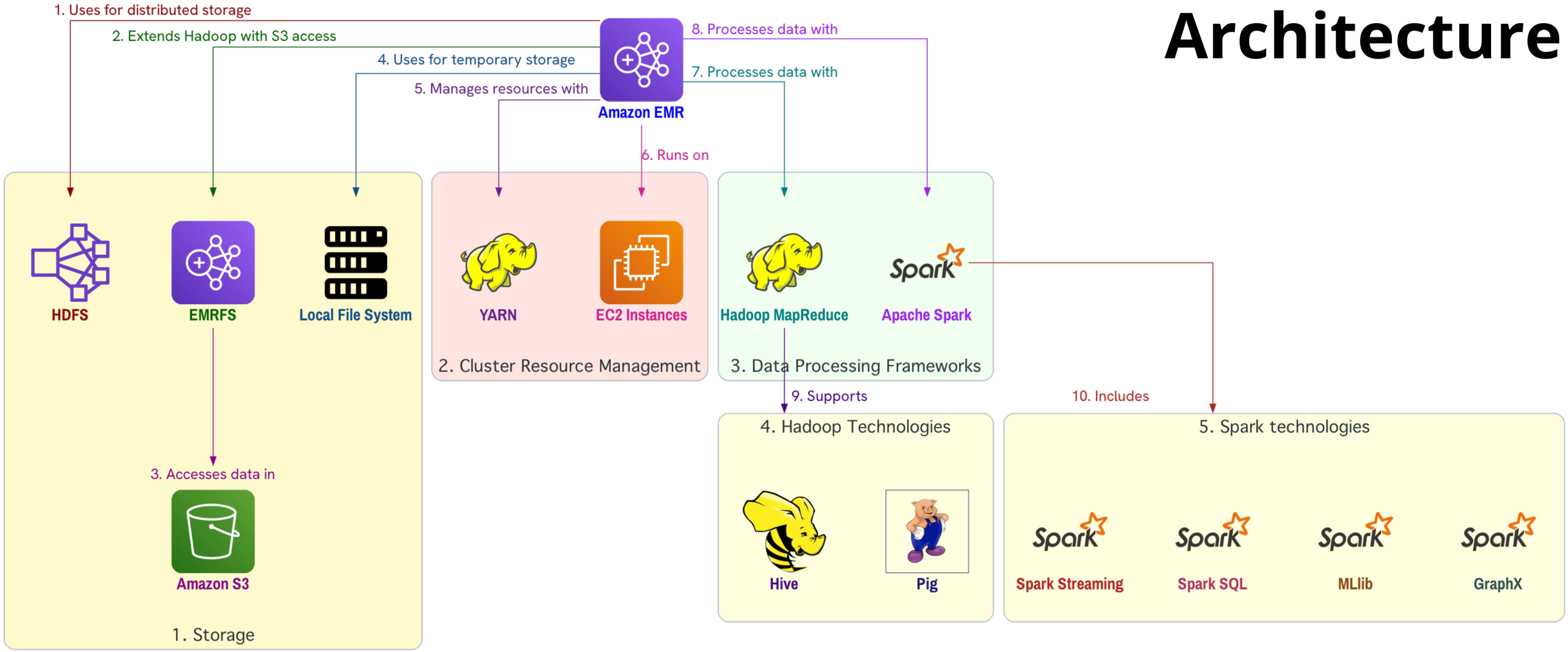
3️⃣ . 🔄 Processing Frameworks: 🐘 MapReduce: 🌐 Open-source distributed computing, 🧩 Simplifies parallel processing; ⚡ Spark: 📊 Directed acyclic graphs, 💾 In-memory caching, 🚀 Faster big data processing; 🔌 Support: 🖥️ High-level interfaces, 🈯 Multiple languages

Architecture



4️⃣ . 🛠️ Applications & Programs: 🐝 Hive: 🗄️ Data warehouse software, 🔍 SQL-like queries, 📊 Data summarization; 🐷 Pig: 🏗️ High-level platform, 🦣 Creates MapReduce programs, 🈯 Pig Latin language; 🌟 Spark Ecosystem: 🔍 Spark SQL, 🌊 Streaming, 🧠 MLlib (machine learning), 🕸️ GraphX (graph processing); 📚 Multiple libraries and languages: 🔄 Flexible data interaction

Architecture



5️⃣ . 🔗 Integration: ☁️ S3: 💰 Cost-effective storage, 🛡️ Durable storage; 💻 EC2: 📈 Scalable compute resources; 💡 Spot Instances: 💰 Cost reduction, 🔄 Processing reliability

Thanks for Watching