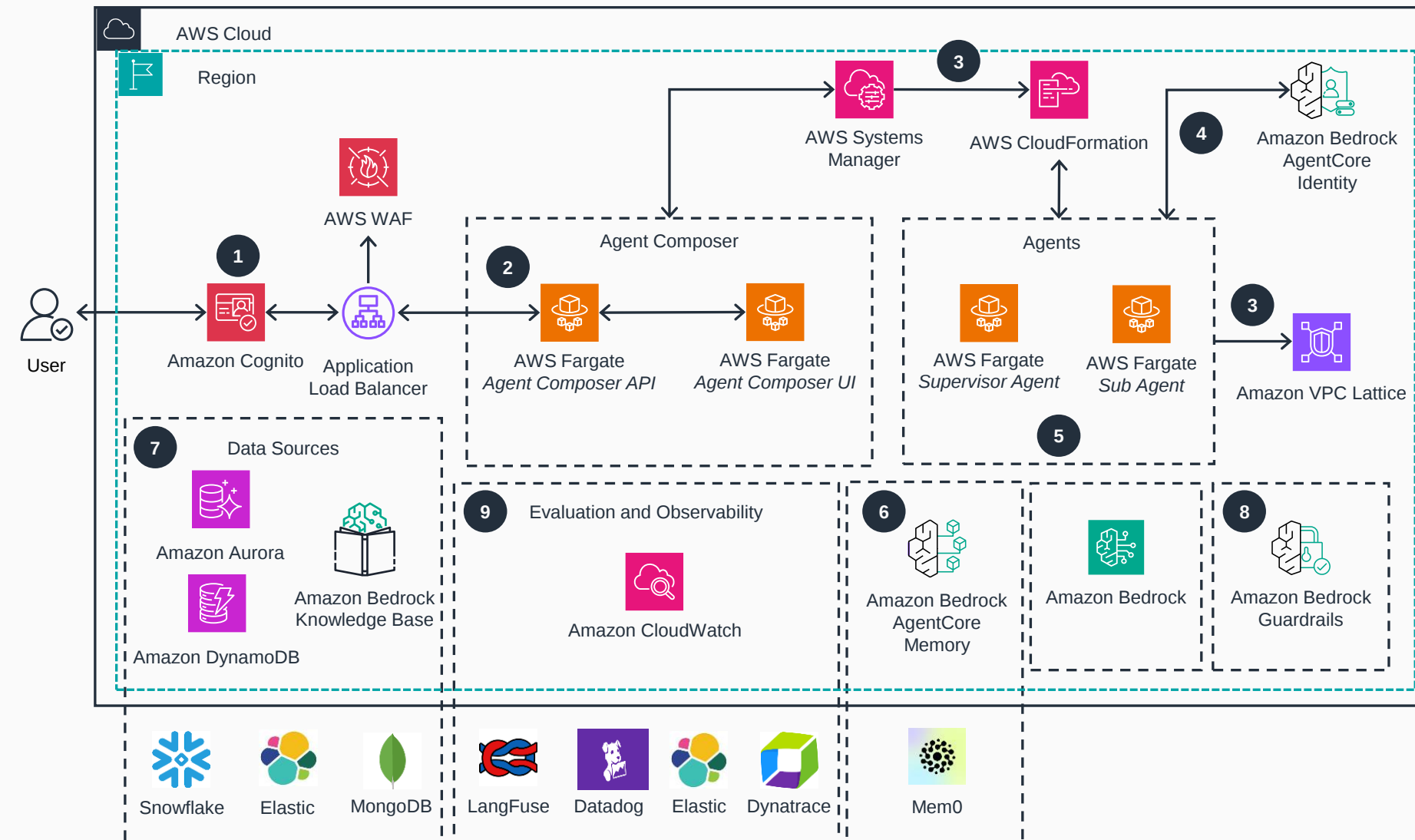


Guidance for No-Code Multi-Agent AI Orchestration on AWS

This architecture demonstrates a no-code, recipe-based tool built on Strands Agents SDK for creating multi-agent AI applications in customer VPCs with existing data platforms and Amazon Bedrock models. This slide shows steps 1-6.

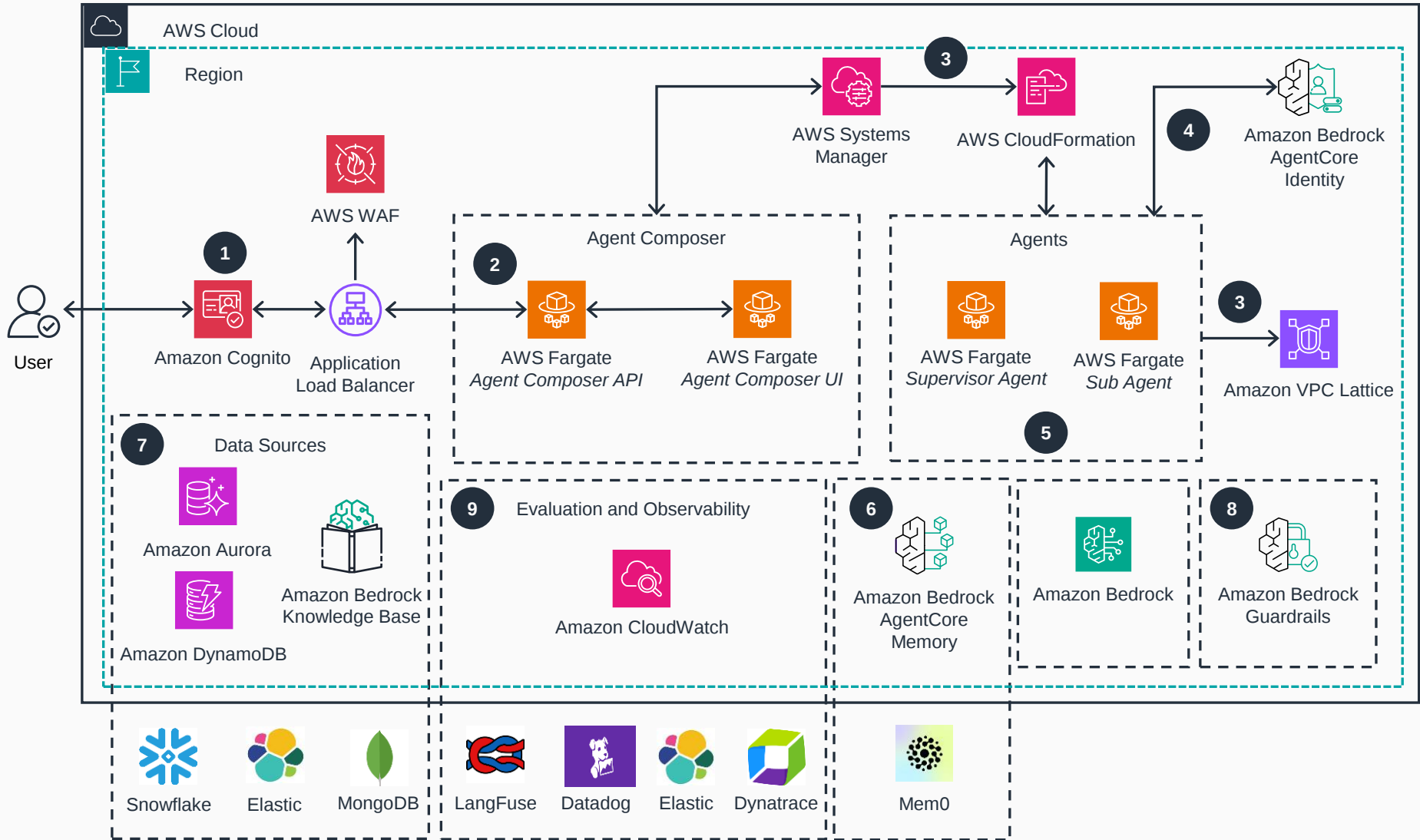


- 1 The user accesses the Agent Composer UI via **Application Load Balancer**. **Amazon Cognito** handles user authentication and authorization. **AWS WAF** provides security protection for OWASP Top 10 security risks at the entry point.
- 2 After authentication, the **Application Load Balancer** routes requests to the Agent Composer API running in **AWS Fargate**, which retrieves existing agent configurations from **AWS Systems Manager** Parameter Store. Users can build new agents or modify existing ones using the Agent Composer UI also running in **AWS Fargate**.
- 3 Once the user clicks Create Agent, the **Agent Composer API** retrieves configurations from **AWS Systems Manager** and triggers **AWS CloudFormation** to deploy a new **AWS Fargate** container for the agent. Once deployment completes, **Amazon VPC Lattice** registers the agent endpoint for multi-agent collaboration.
- 4 **Amazon Bedrock AgentCore** Identity enables AI agents to securely access AWS services and third-party tools, managing both inbound authentication for user-to-agent communication and outbound authentication for agent-to-agent interactions for secure multi-agent workflows.
- 5 The Supervisor Agent running in **Amazon Fargate** acts as the central orchestrator, analyzing user queries using **Amazon Bedrock** foundation models to determine the appropriate specialized agent to handle each request. Each specialized agent runs in dedicated **Amazon Fargate** containers and uses **Amazon Bedrock** foundation models for planning, reasoning, and response generation, with all inter-agent communication facilitated through **Amazon VPC Lattice** service mesh.
- 6 If memory is enabled in the agent configuration, **Amazon Bedrock AgentCore** Memory or Mem0 manages session and long-term memory, providing relevant conversation context to models. The agent retrieves long-term memory from the memory provider (**Amazon Bedrock AgentCore** Memory or Mem0) as specified during initialization and saves conversations as short-term memory.



Guidance for No-Code Multi-Agent AI Orchestration on AWS

This architecture demonstrates is a no-code, recipe-based tool built on strands agents SDK for creating multi-agent AI applications in customer VPCs with existing data platforms and Amazon Bedrock models. This slide shows steps 7-9.



- 7 If an **Amazon Bedrock** Knowledge Base is enabled, agents perform semantic search on configured data sources including AWS services (**Amazon Aurora**, **Amazon DynamoDB**, **Amazon Bedrock** Knowledge Bases) or Independent Software Vendor platforms (Snowflake, Elastic, MongoDB). The search results enhance agent responses with relevant information.
- 8 If guardrails are enabled in the agent configuration, **Amazon Bedrock** Guardrails helps implement safeguards for safety, privacy, and truthfulness safeguards for the generative AI application.
- 9 If observability is enabled, the system pushes logs, metrics, and traces to the configured platform (**Amazon CloudWatch**, Langfuse, Datadog, Dynatrace, Elastic, or other OpenTelemetry-compatible systems). This enables comprehensive monitoring and debugging of agent interactions and performance.