Prompt (im Kontext nach dem Architekturentwurf an ChatGPT 5.1 Thinking):

erstelle mir ein konkretes Beispiel-Setup

Antwort:

Alles klar, ich mache das jetzt konkret – so, als würdet ihr **morgen mit diesem Setup auf Azure starten**. Ich zeige dir:

1. Annahmen & Namenskonvention
2. Netzwerk- & Security-Setup
3. Compute- & KI-Setup (AKS, OpenAI, Agents)
4. Data-Layer-Setup
5. Observability / Security-Setup
6. DevOps & Terraform-Setup (inkl. Beispiel-Snippets)

Ich halte es „konkret, aber noch lesbar“, nicht jede Kleinigkeit auf Ressourcenniveau – eher ein **Blueprint**, den man in Terraform übersetzen kann.

---

1. Annahmen & Namenskonvention

Umgebung:

- Subscription: `sub-ai-prod-001`
- Region: `westeurope`
- Tenant: euer Produktiv-Tenant

Namenskonvention (Beispiele):

- Resource Group: `rg-<scope>-<env>-<region>`
    - z. B. `rg-network-prod-weu`, `rg-aks-prod-weu`
- AKS: `aks-ai-prod-weu`
- VNet: `vnet-hub-prod-weu`, `vnet-aks-prod-weu`, `vnet-data-prod-weu`
- Key Vault: `kv-ai-prod-weu`
- Log Analytics: `law-ai-prod-weu`

Umgebungen:

- `dev`, `test`, `prod` – identische Struktur, andere Größen und ggf. weniger Zonen.

---

2. Netzwerk- & Security-Setup

2.1 Resource Groups

- `rg-network-prod-weu` – VNet, Firewall, Bastion, DDoS, Peerings
- `rg-aks-prod-weu` – AKS, ACR, Managed Identities
- `rg-app-prod-weu` – APIM, Front Door, Application Gateway, Event Hubs, Service Bus
- `rg-data-prod-weu` – SQL, Cosmos, Storage, Data Explorer/Synapse
- `rg-secops-prod-weu` – Log Analytics, Sentinel, Key Vault, Managed Grafana
- `rg-ops-prod-weu` – devOps Hilfsressourcen (Automation, Runbooks, ggf. Jump-VM, Splunk Forwarder etc.)

2.2 Netzwerk-Topologie (Beispiel)

Hub-Spoke-Modell

- Hub-VNet: `vnet-hub-prod-weu` – `10.0.0.0/16`
  - Subnet `snet-azurefw` (Firewall): `10.0.0.0/24`
  - Subnet `snet-bastion`: `10.0.1.0/24`
  - Subnet `snet-shared-services`: `10.0.2.0/24`
- Spoke-VNet AKS: `vnet-aks-prod-weu` – `10.1.0.0/16`
  - Subnet `snet-aks-nodes`: `10.1.0.0/20`
  - Subnet `snet-aks-system`: `10.1.16.0/24`
  - Subnet `snet-aks-user`: `10.1.17.0/24`
- Spoke-VNet Data: `vnet-data-prod-weu` – `10.2.0.0/16`
  - Subnet `snet-data-sql`: `10.2.0.0/24`
  - Subnet `snet-data-analytics`: `10.2.1.0/24`

Komponenten:

- **Azure Firewall** in Hub, alle Spokes via VNet-Peering angebunden
- **DDoS Protection Standard** auf Hub-VNet
- **Azure Bastion** im Hub für SSH/RDP auf Admin-VMs oder Knoten (wenn nötig)
- PaaS-Services (SQL, Storage, Event Hubs, Service Bus, OpenAI, Key Vault) via **Private Endpoints** in `vnet-data-prod-weu` und `vnet-aks-prod-weu`.

---

3. Compute- & KI-Setup

3.1 AKS-Cluster

- Ressource: `aks-ai-prod-weu` in `rg-aks-prod-weu`
- Konfiguration (Beispiel):
    - Kubernetes-Version: `1.29.x` (aktuelle LTS zum Rolloutzeitpunkt)
    - Nodepools:
        - `systempool` (B4ms, 3 Nodes, autoscale 3–10)
        - `gpubatch` (Standard_NC4as_T4_v3, autoscale 0–10) – für Model Serving / featurized Inference
        - `appworkload` (D4as_v5, autoscale 3–20) – AI Agents, LLM-Router, APIs, Frontend
    - Network: **Azure CNI**, IPAM über oben beschriebenes AKS-VNet
    - **AKS-managed AAD Integration** (Azure Entra ID) für Kubernetes-RBAC
    - **OIDC Issuer** für Workload-Identitäten (Federation mit Entra ID, kein Secret-basiertes Zugriffskonzept mehr)

Namespaces (Beispiele):

- `ns-frontend` – UI-Container
- `ns-api` – API & LLM-Router
- `ns-ai-agents` – Agent-Orchestration
- `ns-ml-models` – self-hosted Modelle
- `ns-observability` – Prometheus/Grafana-Agent, OpenTelemetry, etc.

3.2 Container Registry

- Ressource: `acr-ai-prod-weu` in `rg-aks-prod-weu`
- Private Endpoint zu `vnet-aks-prod-weu`
- Defender for Cloud / Defender for Containers Scan auf Images aktiv.

3.3 LLM-Router & Agentic AI

LLM-Router (Beispiel):

- Deployment `llm-router` in `ns-api` (z. B. Python/FastAPI)
- Patterns:
    - Endpunkte: `/v1/chat`, `/v1/completions`, `/v1/tools/<toolname>`

- Routing-Tabelle in ConfigMap/Secret (z. B. `gpt4o` → Azure OpenAI, `internal-llm` → AKS-Model)

Agentic AI Services:

- Deployments `agent-orchestrator`, `retriever-service`, `task-planner` in `ns-ai-agents`
- Kommunikation mit:
  - Azure AI Search (`VECTOR`)
  - SQL/Cosmos für Domänendaten
  - Azure OpenAI / self-hosted Models über interne Services

3.4 Azure OpenAI & Self-hosted Models

- Ressource: `aoai-prod-weu` (Azure OpenAI / Azure AI Foundry Project)
  - Region: `westeurope` oder andere EU-Region, Data Residency „EU only“
  - Bereitgestellte Modelle: `gpt-4.x`, `gpt-4o`, `embedding-3-large` etc.
- Self-hosted Modelle (z. B. Llama, Mistral):
  - Als `Deployment` in `ns-ml-models`
  - Exponiert via `ClusterIP` Services, nur intern erreichbar
  - GPU-Nodepool `gpubatch` zugeordnet.

---

4. Data-Layer-Setup

4.1 SQL, NoSQL, Graph

- **Azure SQL Managed Instance**: `sqlmi-ai-prod-weu` in `rg-data-prod-weu`
  - Nutzung: Referenzdaten, Konfigurationen, Audit-Tabellen, fachliche Daten
  - Always Encrypted / TDE mit CMK aus Key Vault
- **Azure Cosmos DB (Core + Gremlin)**:
  - Account: `cosmos-ai-prod-weu`
  - DBs:
    - `aiapp` (Core API) – Session-Daten, Konfigurationsobjekte, Agent-States
    - `graph` (Gremlin API) – Graph der Entitäten / Policies / Relationen

4.2 Object Storage / Data Lake / Vektorindex

- **Storage Account**: `staiaprodweu` (Data Lake Gen2 aktiviert)

        - Container:
            - `logs-ai` – Langfrist-Logs (immutable)
            - `model-artifacts` – Model-Checkpoints, Eval-Reports
            - `training-data` – z. B. anonymisierte Historik
    - **Azure AI Search**: `aisearch-ai-prod-weu`
        - Index `idx-documents` (klassisch) und `idx-documents-vector` (Vektor)
        - Nutzt Blob Storage als Quelle (Indexers)
        - Dient RAG-Orchestrator als Retrieval-Ziel.

4.3 Messaging & Streaming

- **Event Hubs Namespace**: `evh-ai-prod-weu`
    - Event Hub `eh-ai-telemetry` (Kafka-kompatibler Endpoint) – Streams von Request-/Systemevents
- **Service Bus Namespace**: `sb-ai-prod-weu`
    - Queues: `q-agent-tasks`, `q-agent-results`
    - Topics: `t-notifications`, `t-compliance-events`

4.4 Analytics

- **Azure Data Explorer / Synapse Data Explorer**: `adx-ai-prod-weu`
    - Tabellen: `ai_requests`, `ai_decisions`, `security_events`
- Optional: **Synapse Workspace** für Reporting und DWH-Workloads.

---

5. Observability & Security-Setup

5.1 Logging & Monitoring

- **Log Analytics Workspace**: `law-ai-prod-weu` in `rg-secops-prod-weu`
- Datenquellen:
    - AKS (Container Logs/Insights)
    - APIM, Application Gateway, Front Door
    - Event Hubs, Service Bus, SQL, Storage
    - Azure OpenAI (über Diagnostic Settings, soweit verfügbar)
- **Azure Monitor + Application Insights**:
    - Application Insights Ressource: `appi-ai-prod-weu`

        - Instrumentation in Frontend, API, Agents, Router
        - Distributed Tracing mit Correlation IDs
- **Azure Managed Grafana**:
    - Workspace: `graf-ai-prod-weu`
    - Data Source: Log Analytics, Azure Monitor
    - Dashboards:
        - „Platform Health“ (CPU, Memory, Node Status, Error Rates)
        - „LLM Performance“ (Latenzen, Tokenverbrauch, Rate-Limits)
        - „Compliance & Oversight“ (Anzahl manueller Overrides, abgebrochene Anfragen, Policy-Verstöße)

5.2 SIEM / SOC

- **Microsoft Sentinel**: `sentinel-ai-prod-weu` (auf `law-ai-prod-weu`)
    - Daten-Connectors: Azure Activity, Sign-In Logs, Defender, AKS, APIM, WAF-Logs, Custom Logs von Agents
    - Regeln:
        - Anomalie bei LLM-Nutzung (z. B. plötzlicher Spike an bestimmten Endpunkten)
        - Ungewöhnliche Admin-Logins (PIM, Key Vault, AKS)
- **Splunk**:
    - Entweder via Add-on für Azure Monitor oder Event Hub Export
    - Forwarder in `rg-ops-prod-weu` (VM in `snet-shared-services`)

5.3 Identity, Key Management & Bastion

- **Azure Entra ID**:
    - Security-Gruppen: `sg-aks-admins`, `sg-ai-dev`, `sg-ai-ops`, `sg-soc`
    - App Registrations für Frontend, API, Agents (client credentials / auth code with PKCE)
- **PIM**:
    - Rollen wie `Owner`, `Contributor`, `User Access Administrator`, `AKS Cluster Admin`, `Key Vault Admin` nur via PIM.
- **Key Vault**: `kv-ai-prod-weu`
    - Secrets:
        - DB Connection Strings (wenn nicht komplett MI-basiert)

        - API Keys (z. B. Drittsysteme), Webhook-Secrets
    - Keys:
        - CMK für SQL, Storage, eventuell OpenAI (falls unterstützt)
    - Certificates:
        - TLS-Zertifikate für App Gateway, APIM (öffentlich + privat)
- **Azure Bastion**:
    - Zugriff auf Jump VM (z. B. für Splunk-Forwarder, Spezial-Tools) ohne Public IP.

---

6. DevOps & Terraform-Setup

6.1 Git & Pipelines

- **GitHub Enterprise** Org: `mycompany-ai` (oder Azure DevOps Project `ai-platform`)
- Repositories:
    - `infra-landingzone` – VNet, RGs, Policies, Key Vault, Log Analytics, Firewall
    - `infra-platform` – AKS, ACR, APIM, OpenAI, Data Layer
    - `app-llm-router`, `app-ai-agents`, `app-frontend` – Applikationscode
- Pipelines:
    - `infra-landingzone-ci` / `cd` → Terraform Plan/Apply gegen `dev`/`test`/`prod`
    - `app-*-ci` → Build, Tests, Container-Build, Push nach ACR
    - `app-*-cd` → Deploy nach AKS (Helm)

6.2 Terraform – Beispiel-Snippets (stark vereinfacht)

Backend & Provider (infra-landingzone)

```
terraform {
  required_version = ">= 1.8.0"

  backend "azurerm" {
    resource_group_name  = "rg-tfstate-prod-weu"
    storage_account_name = "sttfstateprodweu"
    container_name       = "tfstate"
    key                  = "landingzone-prod.tfstate"
  }

  required_providers {
    azurerm = {
      source  = "hashicorp/azurerm"
      version = "~> 4.0"
    }
  }
}
```

```
provider "azurerm" {
  features {}
}
```

Beispiel: VNet + Subnet für AKS

```
resource "azurerm_virtual_network" "vnet_aks" {
  name                = "vnet-aks-prod-weu"
  location            = "westeurope"
  resource_group_name = "rg-network-prod-weu"
  address_space       = ["10.1.0.0/16"]
}

resource "azurerm_subnet" "snet_aks_nodes" {
  name                 = "snet-aks-nodes"
  resource_group_name  = azurerm_virtual_network.vnet_aks.resource_group_name
  virtual_network_name = azurerm_virtual_network.vnet_aks.name
  address_prefixes     = ["10.1.0.0/20"]
}
```

Beispiel: AKS-Cluster (gekürzt)

```
resource "azurerm_kubernetes_cluster" "aks_ai" {
  name                = "aks-ai-prod-weu"
  location            = "westeurope"
  resource_group_name = "rg-aks-prod-weu"
  dns_prefix          = "aks-ai-prod"

  sku_tier = "Standard"

  default_node_pool {
    name                = "systempool"
    vm_size             = "Standard_D4as_v5"
    node_count          = 3
    enable_auto_scaling = true
    min_count           = 3
    max_count           = 10
    vnet_subnet_id      = azurerm_subnet.snet_aks_nodes.id
  }

  identity {
    type = "SystemAssigned"
  }

  azure_active_directory_role_based_access_control {
    managed                = true
    admin_group_object_ids = ["<GUID-AKS-Admins-Group>"]
  }

  oidc_issuer_enabled       = true
  workload_identity_enabled = true

  network_profile {
    network_plugin = "azure"
  }
}
```

Beispiel: Log Analytics + Diagnostics

```
resource "azurerm_log_analytics_workspace" "law_ai" {
  name                = "law-ai-prod-weu"
  location            = "westeurope"
```

```
  resource_group_name = "rg-secops-prod-weu"
  sku                 = "PerGB2018"
}

resource "azurerm_monitor_diagnostic_setting" "aks_diag" {
  name                       = "diag-aks-ai-prod"
  target_resource_id         = azurerm_kubernetes_cluster.aks_ai.id
  log_analytics_workspace_id = azurerm_log_analytics_workspace.law_ai.id

  enabled_log {
    category = "kube-apiserver"
  }

  enabled_log {
    category = "kube-audit"
  }

  metric {
    category = "AllMetrics"
  }
}
```

Beispiel: Azure OpenAI (Symbolisch)
(Hinweis: die Terraform-Resource ist je nach Provider-Version ggf. unterschiedlich; pseudomäßig:)

```
resource "azurerm_cognitive_account" "aoai" {
  name                = "aoai-prod-weu"
  location            = "westeurope"
  resource_group_name = "rg-app-prod-weu"
  kind                = "OpenAI"
  sku_name            = "S0"

  custom_subdomain_name = "aoai-prod"

  identity {
    type = "SystemAssigned"
  }
}
```

Wenn du magst, kann ich als nächsten Schritt:

- das Beispiel auf **„nur KI-Kernplattform“** zurechtschneiden (Minimal-Setup),
- oder dir eine **konkrete AKS-Namespace- und Helm-Chart-Struktur** für LLM-Router & Agents vorschlagen.