Estándar de Arquitectura Construcción y Despliegue de MCP (Model Context Protocol) en AWS EKS

Estándar MCP AWS EKS
11/09/2025

Los estándares definidos en este documento han sido establecidos para el cumplimiento en ITAÚ Chile.
Los estándares o patrones pueden ser diferentes a los aplicados en la industria TI.

Historial de Revisiones

Fecha	Versión	Descripción	Autor	Área
11/09/2025	1.0.0	Primer release del estándar de construcción y despliegue de MCP en AWS EKS con KEDA.	COE Enterprise and Solutions Architects Juan Jesus Morales Erick Ibarra Fabian Peña Torres Eduardo Lincopán Francisco Briones	CTO Arquitectura e Infraestructura

Tabla de Contenidos

1. Introducción

Model Context Protocol (MCP) es una tecnología que brinda interoperabilidad, escalabilidad y estandarización para servidores de contexto de modelos de IA, utilizada por las empresas de todo el mundo para acelerar la implementación de soluciones de inteligencia artificial empresarial. FastMCP es una implementación pythónica rápida que permite el desarrollo ágil de servidores MCP mediante abstracción y minimización de código boilerplate.

Las ventajas de implementar MCP con FastMCP en AWS EKS incluyen:

- **Escalabilidad**: Autoscaling event-driven mediante KEDA
- **Flexibilidad**: Despliegue cloud-native en Kubernetes
- **Reducción de costos**: Scale-to-zero y right-sizing automático
- **Menor complejidad operacional**: GitOps y deployment automatizado
- **Seguridad**: Controles de seguridad nativos de Kubernetes y AWS
- **Observabilidad**: Telemetría integrada y monitoreo completo

Este estándar busca definir un marco de uso para la Construcción y Despliegue de servidores MCP utilizando FastMCP en AWS EKS con KEDA, evitando el uso de ECS y ECR, implementando flujos GitOps completos desde GitLab dentro del banco Itaú Chile.

2. Referencias

[1] **Estándar de Herramientas CICD** - Arquitectura Tecnológica (sharepoint.com)

[2] **Procedimiento de validación de arquitectura de soluciones ATI-PRCD-02**

[3] **AWS Well-Architected Framework** - https://aws.amazon.com/architecture/well-architected/

[4] **Model Context Protocol Specification** - https://modelcontextprotocol.io/

[5] **RFC 2119** - https://datatracker.ietf.org/doc/html/rfc2119

[6] **RFC 2616 Hypertext Transfer Protocol (HTTP)** - https://datatracker.ietf.org/doc/html/rfc2616

[7] **FastMCP Documentation** - https://github.com/jlowin/fastmcp

[8] **KEDA Documentation** - https://keda.sh/

[9] **GitOps Principles** - https://opengitops.dev/

[10] **Política CoE Hiperautomatización**

3. Convenciones utilizadas

Las palabras clave del nivel de requisitos "DEBEN", "NO DEBEN", "REQUERIDOS", "DEBERÁN", "NO DEBERÁN", "DEBERÍAN", "NO DEBERÍAN", "RECOMENDADOS", "PUEDEN" y "OPCIONALES" utilizadas en este documento (sin distinción entre mayúsculas y minúsculas) deben interpretarse como se describe en el estándar RFC 2119.

4. Lineamientos para implementación Construcción y Despliegue de MCP

Se describen a continuación los lineamientos que se deben seguir para la construcción y despliegue de servidores MCP utilizando FastMCP en AWS EKS con KEDA dentro del Banco Itaú.

4.1 Lineamientos generales

1. Se requiere aprobación explícita (correo electrónico) del equipo de Arquitectura Empresarial con el detalle del alcance funcional definido para cada iniciativa MCP antes de comenzar la etapa de diseño e implementación.
2. Cualquier incumplimiento en los lineamientos de este documento será considerado como un riesgo pudiendo generar la creación de una OY con el equipo de riesgo operacional.
3. Todas las iniciativas DEBEN contar con la aprobación de CiberSeguridad y Protección del dato antes del despliegue en producción.
4. Se permite la implementación de servidores MCP para casos de uso de colaboradores internos y sistemas de soporte empresarial.
5. NO se permite crear servidores MCP que procesen datos de clientes finales o con impacto directo en sus procesos de negocio sin autorización expresa del CoE de Protección de Datos.
6. NO se permite crear servidores MCP para usuarios externos a la organización.
7. El alcance se limita al uso de AWS EKS como plataforma de orquestación, evitando ECS y ECR como servicios de contenedores.
8. Se DEBE utilizar GitLab Container Registry como registry de imágenes de contenedores, implementando políticas de governance y cleanup automatizado.
9. Todos los despliegues DEBEN seguir el patrón GitOps utilizando pull-based deployment con ArgoCD o Flux.
10. Se DEBE implementar KEDA para autoscaling event-driven, incluyendo capacidad de scale-to-zero para optimización de costos.

4.2 Arquitectura de Seguridad

1. Se debe resguardar que la información involucrada en los flujos de servidores MCP no se vea modificada, adulterada o manipulada de ninguna forma, a excepción de los grupos y usuarios asignados por el negocio, tomando en consideración el análisis de tráfico y monitoreo de los Firewalls Banco (Estándar Seguridad Perimetral, ATI-STDR-16), añadiendo que:
   • Se DEBE contemplar que, para exponer y transmitir al exterior, es preciso hacer uso de protocolo TLS 1.2 o superior
   • Se DEBEN considerar los protocolos de transmisión segura como: HTTPS, gRPC con TLS, y WebSocket Secure (WSS)
   • Todos los ambientes del banco de producción y no productivos (Desarrollo y QA) DEBEN estar separados, tanto física como lógicamente para evitar tráfico entre los ambientes antes mencionados
   • Se DEBE implementar Pod Security Standards (PSS) en modo "restricted" para todos los namespaces de producción

2. En lo que respecta al diseño de soluciones e infraestructura, el solicitante DEBE declarar en sus diseños los puertos TCP/UDP de tráfico de datos, señalización, control y sincronismo, respaldo y administración, para garantizar un entorno seguro (Estándar Tráfico de Aplicaciones y Equipos, ATI-STDR-17):
   • Las conexiones o flujos de red contemplados en el despliegue DEBEN indicar el sentido y origen del tráfico, indicando si el sentido es direccional o bidireccional, además, de señalar el origen y destino
   • Los permisos de conexión entre ambientes o sistemas DEBEN estar orientados a los usuarios específicos en el origen y destino (Por medio de la gestión de identidades del Banco)
   • Se DEBE implementar Network Policies de Kubernetes para microsegmentación de tráfico entre pods
   • Se tiene prohibido el envío de Log's que incluyan datos sensibles, tales como, por ejemplo, claves, llaves, secretos, usuarios entre otros que puedan ser usados para acceder o llamar a algún flujo de la suite a implementar
3. **Gestión de Secretos y Credenciales**:
   • Se DEBE utilizar AWS Secrets Manager integrado con Secrets Store CSI Driver para la gestión de secretos en Kubernetes
   • Las credenciales DEBEN rotarse automáticamente según políticas definidas (mínimo cada 90 días)
   • Se DEBE implementar RBAC granular para acceso a secretos por namespace y service account
4. **Firmado y Verificación de Imágenes**:
   • Todas las imágenes de contenedores DEBEN ser firmadas digitalmente utilizando Cosign
   • Se DEBE implementar admission controllers para verificar firmas antes del despliegue
   • Las imágenes DEBEN pasar escaneo de vulnerabilidades con severity máximo "Medium"

4.3 Arquitectura de Datos

1. NO se permite, sin la autorización expresa del equipo de protección del dato, Seguridad TI y Ciberseguridad, el uso de datos de clientes en servidores MCP.
2. NO se permite el almacenamiento de datos sensibles en bases de datos no homologadas bajo el CoE de Arquitectura de datos sin previa validación con el CoE.
3. Se establece como una herramienta del tipo consumidor de datos, los consumidores de datos que pueden generar analítica para su propio uso y/o compartir con otros consumidores de datos. NO se permite la integración con soluciones del tipo productoras de datos de Gobierno TI y/o Gobierno Compartido.
4. Se DEBE considerar las cuotas de recursos asignadas por tipo de licencia AWS para todos los componentes de la solución (EKS, storage, compute).
5. Se DEBE generar un plan para monitorear las aplicaciones MCP desarrolladas en la organización por la Comunidad y/o Unidad de Negocio. El monitoreo DEBE incluir métricas de performance, disponibilidad y utilización de recursos.
6. Se DEBE considerar la dependencia del soporte del proveedor sobre posibles incidencias relacionadas con FastMCP y componentes de la stack.

7. NO se deben generar accesos hacia los repositorios de datos fuera de la plataforma empresarial autorizada.
8. Se DEBE definir el ciclo de vida de los datos que se traten cualquiera sea su clasificación, incluyendo políticas de retención y eliminación.
9. Se DEBE tener una separación física y/o lógica entre las capas de datos dentro de la aplicación y los distintos usuarios que la utilicen mediante namespaces de Kubernetes y Network Policies.
10. **Persistencia de Datos**:
    • Se DEBE utilizar AWS EBS CSI Driver para volúmenes persistentes
    • Los datos DEBEN estar encriptados en reposo utilizando AWS KMS
    • Se DEBE implementar backup automatizado con retention de 30 días para desarrollo y 1 año para producción
11. En caso de existir un caso que no esté abordado con los lineamientos y consideraciones anteriores se deberá revisar y validar la iniciativa o problemática con el CoE Arq. De datos.

4.4 Arquitectura de Soluciones

1. Cada iniciativa será catalogada con un tallaje según Procedimiento de validación de arquitectura de soluciones ATI-PRCD-02 y DEBE ser aprobada por el CoE Experto de Arquitectura, en caso de ser necesario, teniendo potestad el equipo de arquitectura empresarial de solicitar su presentación ante el CTO de Arquitectura.
2. NO debe interactuar con plataformas obsoletas definidas por el CoE de Arquitectura tales como: webmonitor, txserver, modyo, facephi.
3. Para dar inicio al diseño de la solución se DEBE tener cerrado la(s) historia(s) de usuario (HU) con criterios de aceptación definidos.
4. La Comunidad y/o Unidad de Negocio DEBE trabajar en una propuesta de solución para cada historia de usuario; es responsable de la generación de la primera versión del diseño de la solución, la cual DEBE contar con un nivel de completitud del 80% como mínimo.
5. El diseño DEBE cubrir todos los estándares de arquitectura que se encuentran definidos.
6. **Estándares específicos para MCP**:
    • Se DEBE utilizar FastMCP como framework de desarrollo para servidores MCP
    • Los servidores MCP DEBEN implementar health checks (/health y /ready endpoints)
    • Se DEBE implementar graceful shutdown para manejo correcto de terminación de pods
    • Los servidores MCP DEBEN exponer métricas en formato Prometheus en el endpoint /metrics
7. **Patrones de Diseño**:
    • Se DEBE seguir el patrón de microservicios con separación de responsabilidades por dominio
    • Se RECOMIENDA implementar Circuit Breaker pattern para llamadas a servicios externos
    • Se DEBE implementar retry policies con exponential backoff

4.5 Arquitectura de Infraestructura

1. **Ambientes**:
   1.1. **Desarrollo y Pruebas**: Se DEBE considerar tener ambientes separados para desarrollo y pruebas en clusters EKS independientes. Esto evita que los cambios en desarrollo afecten el ambiente de producción. Así, los desarrolladores pueden experimentar y probar sin riesgos.
   1.2. **Producción**: El ambiente de producción DEBE estar bien protegido en un cluster EKS dedicado. Solo los cambios aprobados y probados DEBEN implementarse aquí para asegurar la estabilidad y seguridad del sistema.
   1.3. **Configuración de Clusters EKS**:
   • Control Plane DEBE ser managed por AWS con logging habilitado
   • Worker Nodes DEBEN utilizar Managed Node Groups distribuidos en múltiples AZs
   • Se DEBE habilitar VPC CNI para networking nativo de pods
   • Se DEBE configurar EBS CSI Driver para almacenamiento persistente
2. **Gobernanza**:
   2.1. **Políticas de Seguridad**: Se DEBE definir quién tiene permiso para crear, modificar y eliminar aplicaciones MCP y flujos KEDA. Esto ayuda a mantener el control y la seguridad dentro de la plataforma mediante RBAC de Kubernetes.
   2.2. **Monitoreo y Auditoría**: Se DEBE implementar herramientas de monitoreo y auditoría permite rastrear todas las actividades y cambios. Esto es vital para detectar y solucionar problemas rápidamente mediante:
   • Prometheus + Grafana para métricas
   • ELK Stack para logs centralizados
   • Jaeger para distributed tracing
   • AWS CloudTrail para audit trail de API calls
3. **Diseño Arquitectónico**:
   3.1. **Modularidad**: Se DEBEN diseñar servidores MCP de manera modular facilita el mantenimiento y la escalabilidad. Cada servidor puede desarrollarse y actualizarse independientemente.
   3.2. **Reutilización de Componentes**: Se DEBE utilizar Helm Charts y templates reutilizables que aceleren el despliegue y aseguren la consistencia en toda la aplicación. Esto también reduce el esfuerzo de desarrollo a largo plazo.
   3.3. **KEDA Configuration**: Se DEBE configurar KEDA para autoscaling basado en eventos:
   • ScaledObjects para definir triggers de scaling
   • Scale-to-zero capability para optimización de costos
   • Múltiples trigger sources (SQS, Prometheus, HTTP, Cron)
4. **Administración**:
   4.1. **Gestión de Usuarios**: Se DEBE definir roles y permisos adecuados para los usuarios mediante RBAC de Kubernetes. Esto asegura que cada persona tenga acceso solo a las funciones necesarias para su trabajo.
   4.2. **Automatización**: Se DEBE utilizar GitOps workflows automatizados para deployment que asegura la consistencia y eficiencia en los procesos. Esto también libera tiempo para que el equipo se enfoque en tareas más estratégicas.

4.6 Arquitectura de Integración

1. NO se permite integración directa con ecosistema de aplicaciones Itaú, APIs bancos, Base de datos de comunidades o de Gobierno compartido sin autorización expresa del CoE de Integración.
2. NO se permite el uso y creación de conectores que no estén homologados por el CoE de Arquitectura, salvo los requeridos para la autenticación y autorización del acceso a la plataforma y los aplicativos construidos previa validación en la etapa de diseño.
3. El despliegue entre ambientes se DEBE realizar con aprobación de Transición TI mediante ticket Jira y se DEBE agregar el certificado ADA correspondiente.
4. Para el versionamiento y control del código se DEBE utilizar la herramienta estándar banco, para mayores detalles revisar el estándar ATI-STDR-36 Estándar de Herramientas CICD.
5. **Integración con GitLab**:
   - Se DEBE utilizar GitLab Container Registry como registry de imágenes
   - Se DEBE configurar GitLab Runners en modo Kubernetes executor
   - Se DEBE implementar GitLab CI/CD pipelines con stages de seguridad y calidad
6. **Service Mesh Integration**:
   - Se RECOMIENDA utilizar Istio o Linkerd para service-to-service communication
   - Se DEBE implementar mTLS entre servicios MCP
   - Se DEBE configurar traffic policies y circuit breakers
7. **API Gateway Pattern**:
   - Se DEBE utilizar ingress controllers (NGINX o ALB) para exposición de servicios
   - Se DEBE implementar rate limiting y authentication en el API Gateway
   - Se DEBE configurar SSL/TLS termination en el ingress

4.7 Organización de carpetas y archivos en el sistema de control de versiones establecido

La definición de la estructura de directorios en el sistema de control de versiones establecido se realizará bajo kebab-case (palabras en minúscula separadas por un guion) y todos los desarrollos estarán bajo el directorio raíz **mcp-servers**

Estructura para Código Fuente MCP:
• **Path**: `mcp-servers/<identificador-inventario-aplicaciones>/<nombre-servidor-mcp>`
• **Ejemplo**: `mcp-servers/ai-platform/customer-support-mcp`

Estructura para Configuraciones Kubernetes:
• **Path**: `mcp-infrastructure/<environment>/<cluster-name>/<namespace>`
• **Ejemplo**: `mcp-infrastructure/production/eks-prod-us-east-1/fastmcp-prod`

Estructura para Helm Charts:
• **Path**: `mcp-charts/<chart-name>`
• **Ejemplo**: `mcp-charts/fastmcp-server`

Estructura estándar de repositorio MCP:

```
mcp-servers/ai-platform/customer-support-mcp/
├── src/
```

```
│   ├── main.py
│   ├── tools/
│   ├── resources/
│   └── requirements.txt
├── tests/
│   ├── unit/
│   └── integration/
├── docker/
│   └── Dockerfile
├── k8s/
│   ├── base/
│   ├── overlays/
│   │   ├── dev/
│   │   ├── staging/
│   └── └── prod/
├── .gitlab-ci.yml
├── README.md
└── CHANGELOG.md
```

Convenciones de Naming:
• Branches: `feature/mcp-nueva-funcionalidad`, `hotfix/mcp-correccion-critica`
• Tags: `v1.0.0`, `v1.0.1-rc1`
• Docker Images: `registry.gitlab.com/itau/mcp-servers/customer-support:v1.0.0`

5. Reviews

Fecha	Versión	Descripción	Autor	Reviewers
11/09/2025	1.0.0	Primer release del estándar de construcción y despliegue de MCP en AWS EKS con KEDA.	Juan Jesus Morales, Erick Ibarra, Fabian Peña Torres, Eduardo Lincopán, Francisco Briones	Diego Solis, Mauricio Sepulveda, Adrian Garcia, Oscar Vega, Carlos Rodriguez (CoE Security), Maria Gonzalez (CoE Data)

6. Anexos

Anexo A: Ejemplo de Pipeline GitLab CI/CD

```
stages:
  - validate
  - security
  - build
  - test
  - package
  - deploy-dev
  - deploy-staging
  - deploy-prod

variables:
  REGISTRY: $CI_REGISTRY
  IMAGE_NAME: $CI_REGISTRY_IMAGE/fastmcp-server
  KUBE_NAMESPACE: fastmcp-${CI_ENVIRONMENT_NAME}
```

```
validate:
  stage: validate
  script:
    - black --check src/
    - flake8 src/
    - mypy src/

security:
  stage: security
  script:
    - bandit -r src/
    - safety check
    - trivy fs .

build:
  stage: build
  script:
    - docker build -t $IMAGE_NAME:$CI_COMMIT_SHA .
    - docker push $IMAGE_NAME:$CI_COMMIT_SHA
```

Anexo B: Ejemplo de ScaledObject KEDA

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: fastmcp-scaler
  namespace: fastmcp-prod
spec:
  scaleTargetRef:
    name: fastmcp-deployment
  minReplicaCount: 0
  maxReplicaCount: 20
  triggers:
  - type: aws-sqs-queue
    metadata:
      queueURL: https://sqs.us-east-1.amazonaws.com/123456789/mcp-requests
      queueLength: "5"
      awsRegion: us-east-1
    authenticationRef:
      name: keda-aws-credentials
  - type: prometheus
    metadata:
      serverAddress: http://prometheus.monitoring.svc.cluster.local:9090
      metricName: mcp_requests_rate
      threshold: '10'
      query: rate(fastmcp_requests_total[1m])
```

Anexo C: Checklist de Cumplimiento

Pre-despliegue:

- [ ] Aprobación explicita del CoE de Arquitectura Empresarial
- [ ] Aprobación de CiberSeguridad y Protección del dato
- [ ] Historias de usuario cerradas con criterios de aceptación
- [ ] Diseño de solución con completitud mínima 80%
- [ ] Definición de puertos TCP/UDP y flujos de red
- [ ] Configuración RBAC y Network Policies
- [ ] Implementación de health checks y métricas
- [ ] Escaneo de vulnerabilidades passed (max severity: Medium)
- [ ] Firmado digital de imágenes implementado
- [ ] Backup y disaster recovery plan definido

Post-despliegue:

- [ ] Monitoreo y alerting configurado
- [ ] Logs centralizados funcionando
- [ ] KEDA autoscaling validado
- [ ] GitOps sync funcionando correctamente
- [ ] Certificado ADA agregado al ticket de despliegue

Los estándares definidos en este documento han sido establecidos para el cumplimiento en ITAÚ Chile.