Agnoda 4 temel yapı var: flags, tools, agent, workflow

- **Flags** → modelin zekasına güven
- **Tools** → modelin dışında özelleştirme
- **Agents** → tools'u "akıllı" hale getirme
- **Workflows** → çoklu adımları birleştirme ve yönetme

Flags:

- Flags, Agno'da genelde **ayar (config)** veya **davranış kontrolü** için kullanılan küçük parametrelerdir
- En alt seviye ve en basit katman.
- **Ne işe yarar:** Modelin `knowledge`, `tools` gibi kaynaklara erişip erişemeyeceğini veya hangi metadata alanlarını kullanabileceğini belirler.
- **Özelleştirme:** Neredeyse yok. Modelin kendi zekasına güveniyorsun, iş mantığını değiştiremiyorsun.
- **Kullanım amacı**:
    - Framework'ün bazı özelliklerini açıp kapamak.
    - Deneysel özellikleri kontrol etmek.
    - Logging, cache, memory kullanımı gibi davranışları yönetmek.
- **Kullanım örneği:**
    - `search_knowledge=True` → agent vektör tabanlı DB'de arama yapabilir
    - `valid_metadata_filters={"application_id"}` → sadece bu metadata'ya göre filtre uygular

Tools:

Tools, **agent'in kullanabileceği yetenekler**dir.

- Fonksiyonellik katmanı. Modelin dışında **özelleştirilmiş fonksiyonlar ve iş mantığı** ekleyebileceğin yer.
- **Ne işe yarar:** Modelin doğal zekasını kullanmak yerine, özel işlemleri (hesaplama, veri çekme, dosya okuma, custom search) kendin yazabilirsin.

- **Özelleştirme:** Çok yüksek. Instructions, few-shot örnekleri, davranış kuralları vs. Eklenebilir.
- **Kullanım amacı**:
    - Dış sistemlerle iletişim (SQL query çalıştırma, dosya yazma, API çağrısı).
    - Çekirdek işlevler (ör. `DuckDuckGoTools` → web araması).
    - Kendi özel fonksiyonlarını agent'e eklemek.
- **Kullanım örneği:**
    - `KnowledgeTools` → semantic search + think + analyze adımları eklemek
    - `MLXTranscribeTools` → model yerine ses dosyasını transkripte eden custom fonksiyon

Agents:

- Tools ve Flags'i birleştirip **akıllı bir karar mekanizması ekleyen katman**.
- **Ne işe yarar:**
    - Agent, Tools'u çağırabilir, Flags'ı dikkate alabilir ve karar verebilir.
    - İçinde "düşünme döngüsü" (think → search → analyze) veya multi-step planlama olabilir.
- **Özelleştirme:** Orta-yüksek. Agent'in davranışını kontrol edebilirsin, ama agent genellikle Tools ve Flags üzerine kurulur.
- **Kullanım örneği:**
    - `SemanticSearchAgent` → KnowledgeTools ile semantic search yapar
    - Agent, session memory ile birden fazla query'i birleştirip optimize edebilir
    - Agno agentları session-state veya kısa süreli hafıza tutabilir. (Bir agent, kullanıcının önceki sorgularını hatırlayarak aramalarını optimize edebilir.)
    -

Workflows

- En üst katman ve **çok adımlı süreçlerin yönetimi**.
- **Ne işe yarar:**
    - Birden fazla agent'ı veya aracı bir araya getirir
    - Adımları sıraya koyar, koşullara göre yönlendirme yapar
    - "Pipeline" gibi düşünülebilir; örneğin:
        - Ses dosyasını transkripte et (`MLXTranscribeTools`)
        - Transkripti semantic search ile sorgula (`KnowledgeTools`)

        - ♣ Analiz et ve özetle (`Agent`)
- **Özelleştirme:** Çok yüksek, çünkü tüm agent'lar ve tools'un davranışını bir workflow içinde yönetebilirsin.
- **Kullanım amacı**:
    - Karmaşık iş akışlarını yönetmek (örn. önce SQL parse → sonra DB search → sonra semantic search → sonra format).
    - Router ile hangi adımların çalışacağına karar vermek.
    - StepInput ve StepOutput ile adımlar arasında veri taşımak.
- **Özellik**:
    - `Step` nesneleri içerir.
    - `Router` ile dallanma yapabilir.
    - Tekrar kullanılabilir süreçler oluşturur.
- **Kullanım örneği:**
    - Otomatik CV işleme workflow'u →
        - ♣ Ajan transkripti çıkarır
        - ♣ Semantic search ile veri eşleştirme
        - ♣ Özetleme ve skor hesaplama