All scopes, All projects	Orchestral Layer	**Melody** Able to "play" composite types Encapsulate operational systems Plays along side accompaniment Root endpoint of composed trees	**Performance** Play to specific audience instance Instruments come and go Tune according to ensemble Hides low/mid level concerns	**Orchestra** Harmonize multiple instruments Normalize intensity by audience Keep tempo in check Tune pitch toward natural "chords"
Intranet, Platform Staging, Platform Production, specific projects	Ensemble Layer	**Orchestra** In-Context Instrument Start / Stop Decides instrument priorities Perform in different conditions A troupe of compute performers	**Instrument** Can be assigned abstract tasks Add instrument to match work Tune according to ensemble Hides low/mid level concerns	**Rythm & Sound** Harmonize multiple instruments Normalize intensity by audience Keep tempo in check Tune pitch toward natural "chords"
Edge nodes, Component Nodes, Data Nodes,..	System Layer	**Instrument** Able to "play" composite types Encapsulate operational systems Constrains requirements* Root endpoint of composed trees	**Composite Types** Collection-level Operations Naturally Produce a DSL lang Manage underying resources Group storage/transport needs	**Rythm & Sound** Aggregates regular status checks Tracks frequency of resource use Normalize frequency to aspect ratios Reveal pitch, intensity and tempo
Details Layout, Admin Layout, Collection Layout, Dynamic Layout..	Presentation Layer	**Layout / Composite Type** Typed, Purposeful Presentations Themes & Styles Audience-Oriented Resources Presentation Isolation	**Composition** Resource Resolver System Level Scopes Interface Service Cache and Datapoint	**Components** Settings & Props Action Module Templates Presentation Logic
Listing Dashboard, Leads Dashboard, AskAskowitz.com/about,..	Composition Layer	**Composition** Component Aggregation DOM Reflection Prepublish/Publish Reveals Subscope	**Components** Data Aggregation Data Bindings Data Access Business Logic	**Service** Attaches to Scopes Exposes Commands/Operations Communication Exchanges Access via Scoped Components
Listing, Product, Comment, Thumbnail Card, Hero Unit, Banner..	Component Layer	**Component** Represent Human Concept Encapsulate Dependencies Resource-Oriented Extensible, Hookable, Bindable	**Datamaps & Bindings** Dependency Definition Enable Map / Reduce Support Automation Links Presentation To System	**Instances & Templates** Ephemeral Contexts Forward/Backward Tracing Simple Module Creation Emulate MVC Workflows
MariaDB, Mongo, APIs, JSON Files..	Data Layer	**Datamaps & Bindings** Generate "select" criteria Import/Export Automation Connect sources to modules Translates system to presentation	**Dataset** Abstracts queries Provides container interface Common transport format Links Presentation To System	**Models** Auto Generated Save/Load + Hooks Simple Module Creation Emulate MVC Workflows
STRINGs, INTs, STRUCTUREs..	Render Layer	**Streams** A branching sequence of iteration Flow in one direction at a time Generally input or ouptut Media Format-agnostic	**Renderable Streams** Is a media format as a primitive Technically, all info is renderable Nodes with branch nodes Basic data building block	**Bound (smart) Nodes** Routes primatives to new structure Also Media-Format Agnostic Driver of raw but abstract streams Contains Binding in a Renderable
CPU / RAM / OS / NETWORK	Work Layer	**Streams** Literal Media going in or out Each streamed item is typed Multiplexing Friendly, Parallel work Combines resource and action	**Work** Actual operations on streams Compute, action or transform Could be human, machine or logic Can use non-Approach units	**Transport** Streams moved in/out of a process Deliver to audience at terminals Pipe from work to work internally Implicit, but lag & volume matter

*Just as any trumpet player may play any trumpet: any host or virtual machine may run an instrument if it has the proper qualifications.

For example, an "edge" instrument may require
- 4 GB RAM
- 10GB disk.
- PHP
- Redis