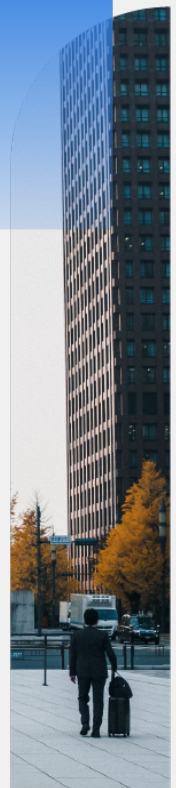


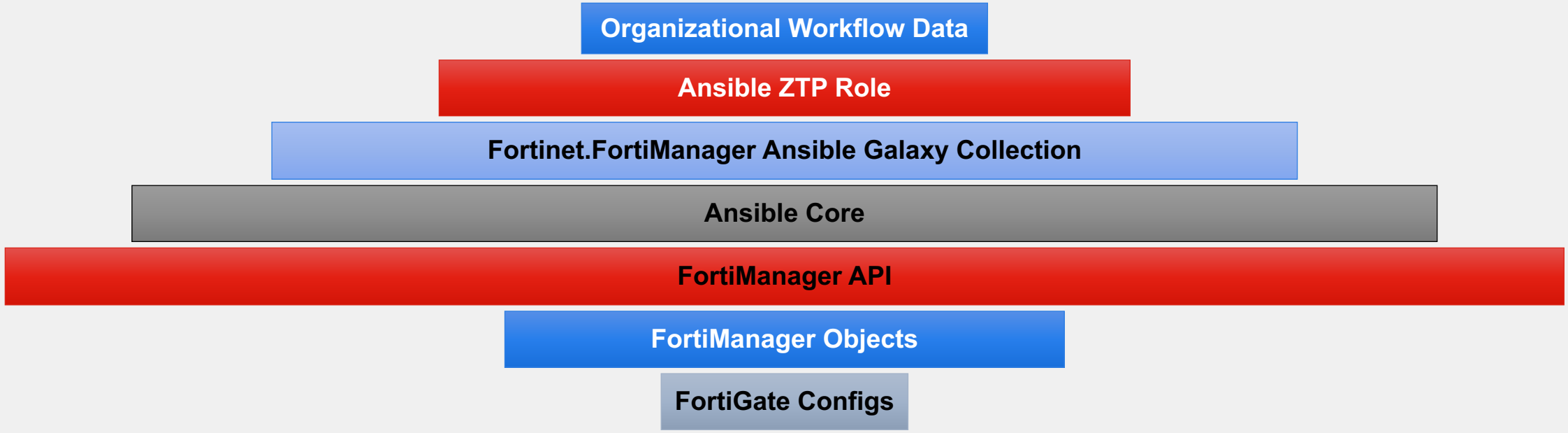
FortiGate Ansible Zero Touch Provisioning (FA-ZTP)

Luke Weighall

CSE DevOps

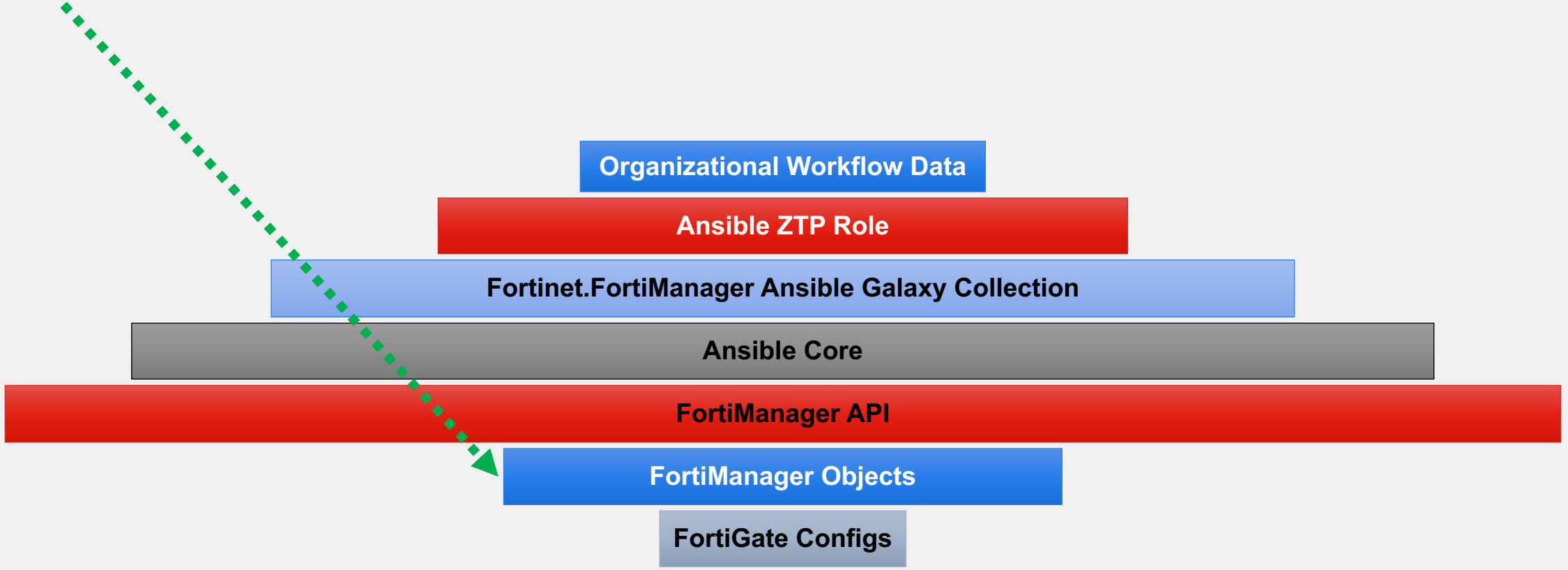
Introduction

- Zero-Touch Provisioning is a data problem.
- Ansible provides an abstraction layer for normalizing Organizational workflow data.
- We are going to explore these layers to better understand the problem at hand.

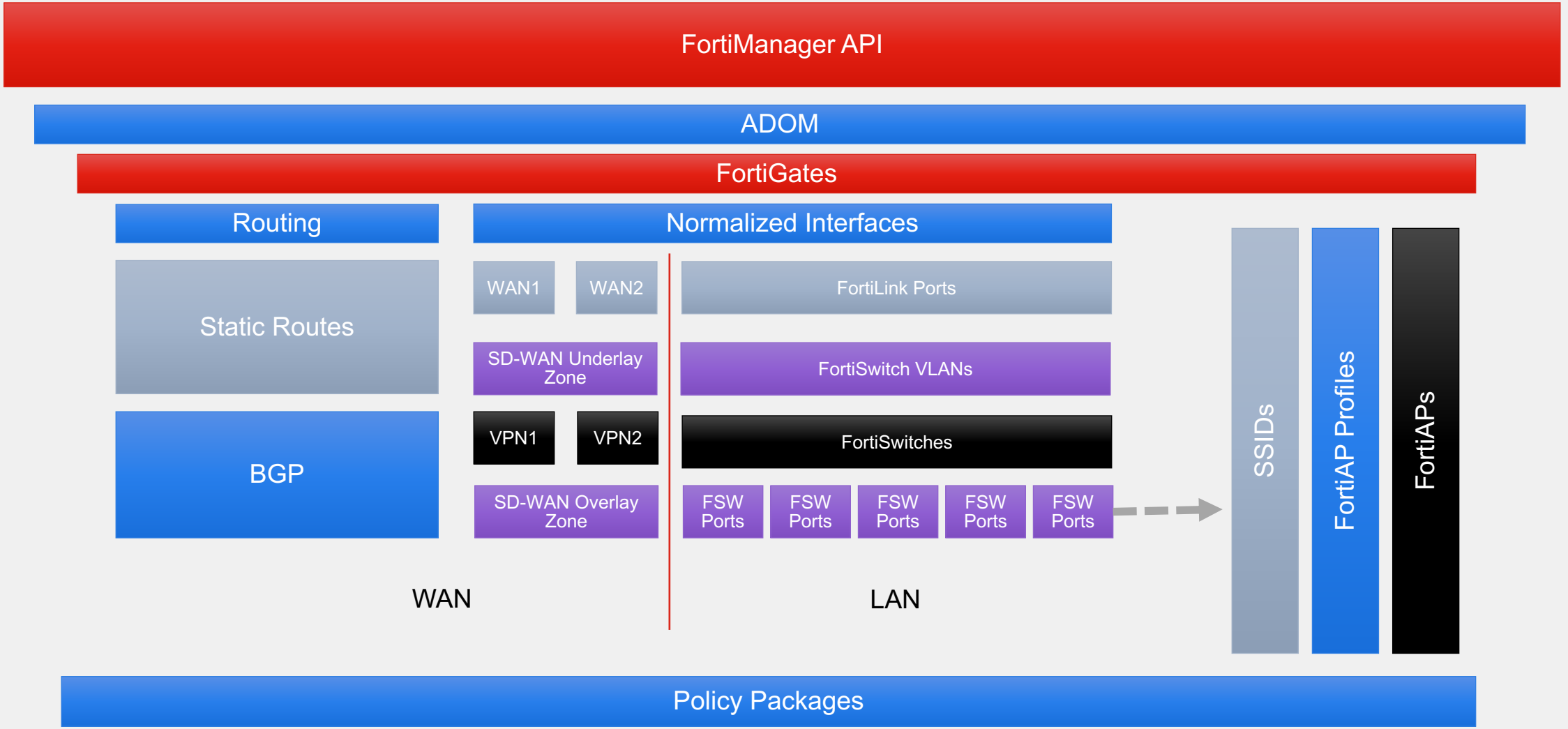


Introduction

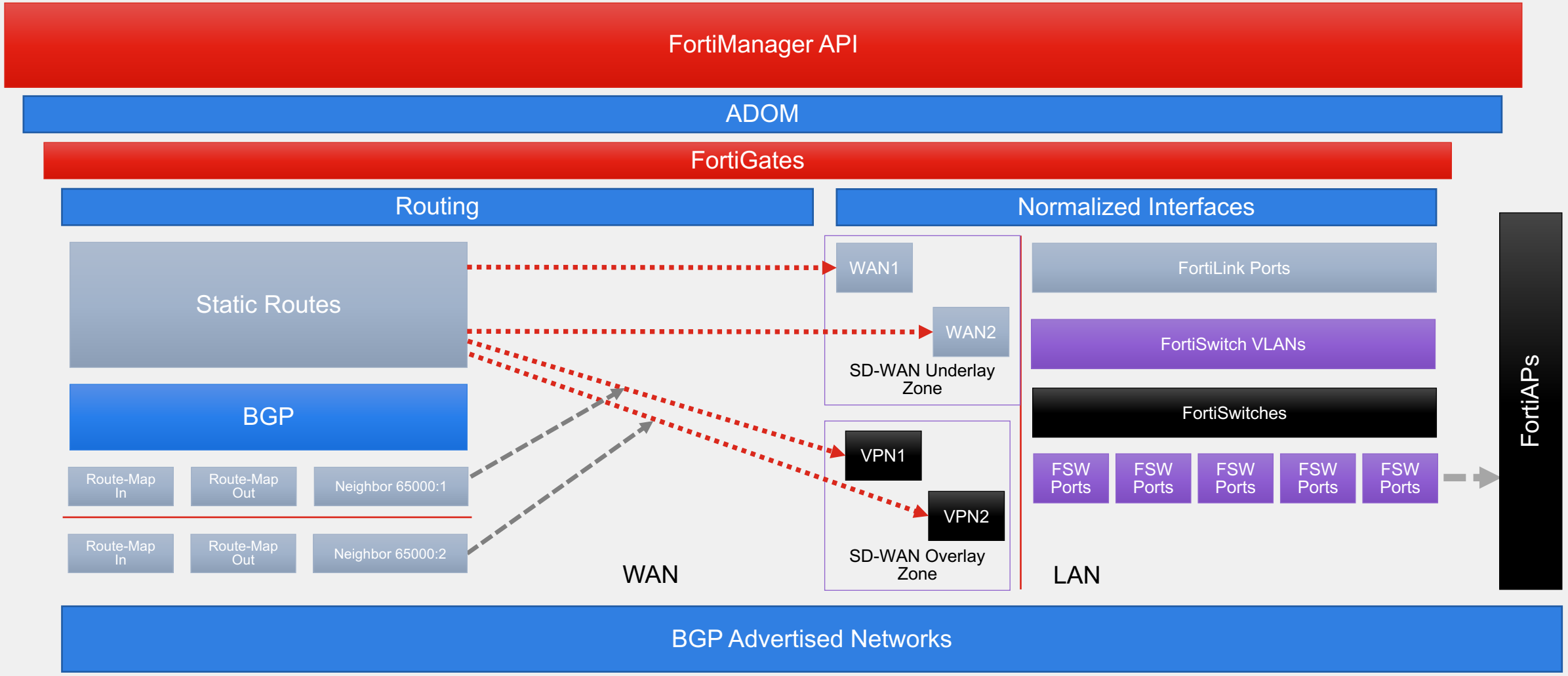
- First we must understand what it takes to deploy a branch FortiGate.
- Let’s explore the FortiManager Objects layer.



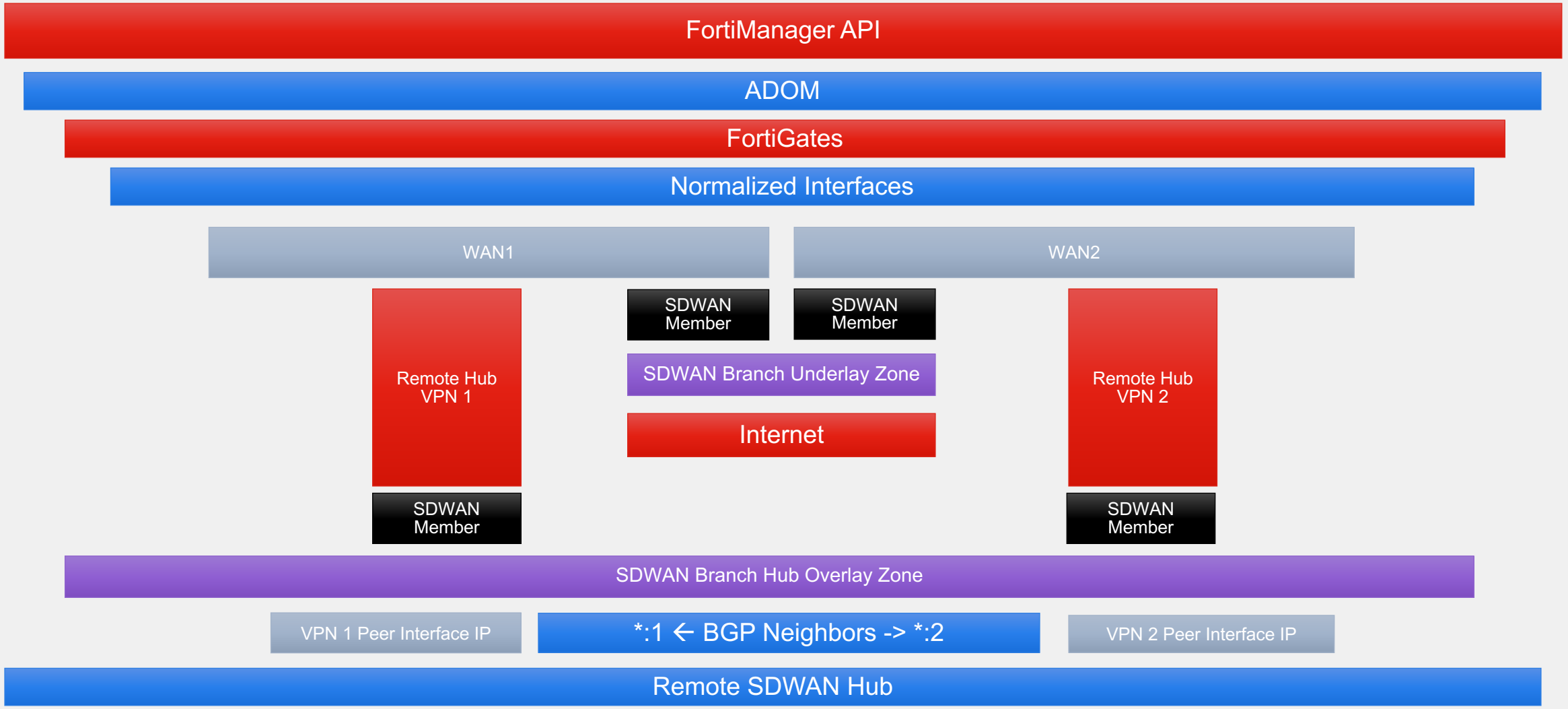
FortiManager Branch ZTP Object Layers



Routing Layer

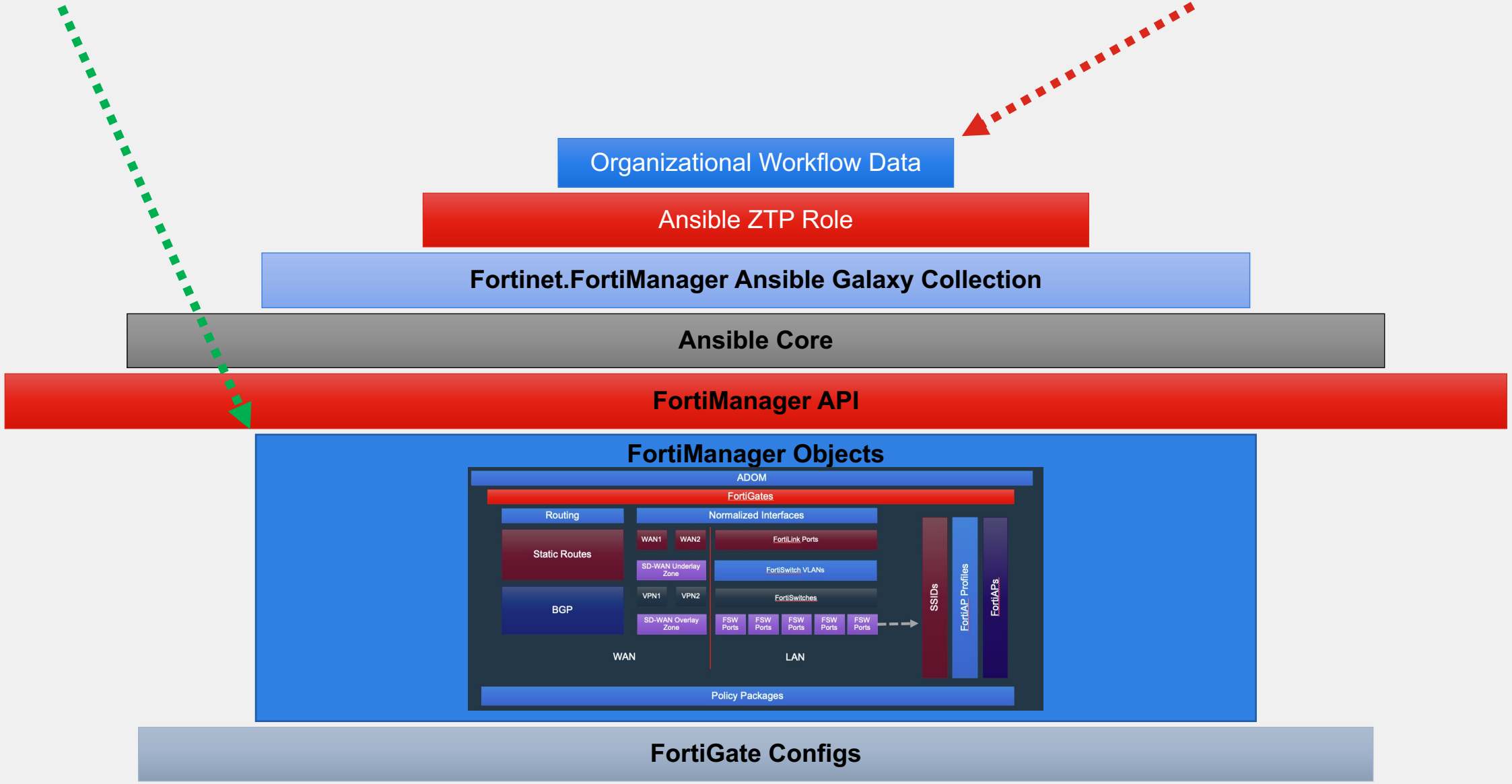


SDWAN Layer

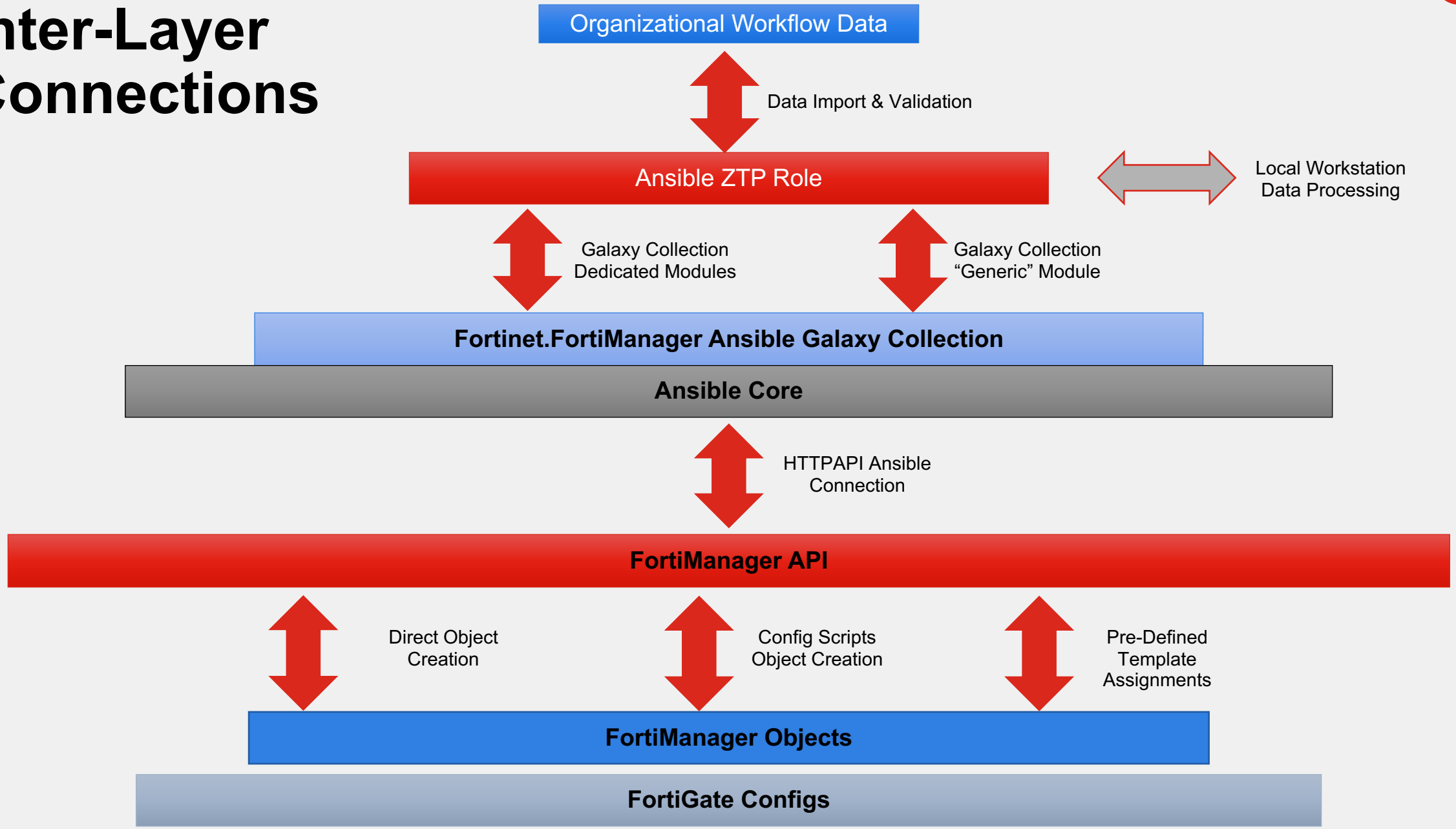


So how do we get here?

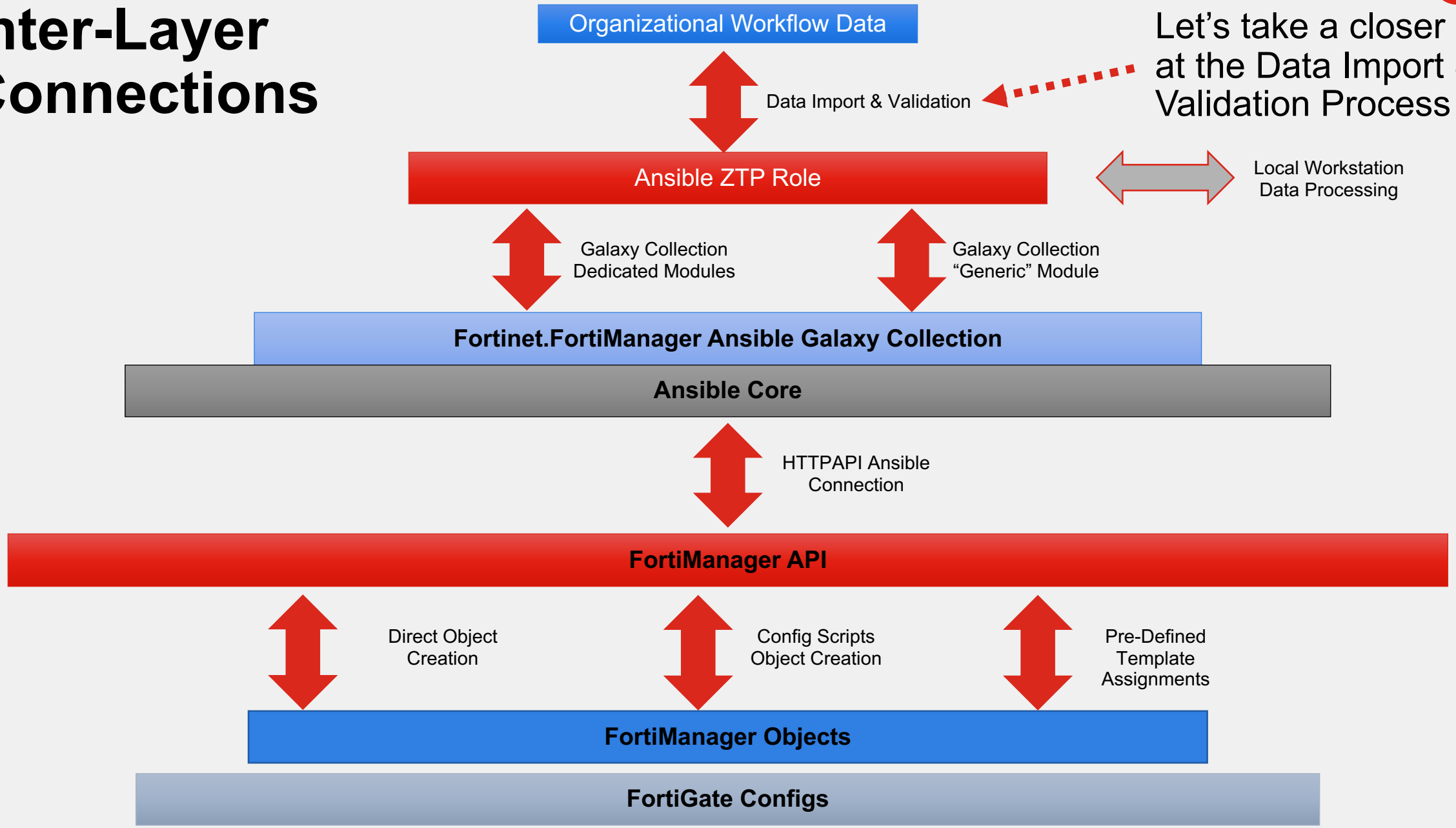
From Here?



Inter-Layer Connections



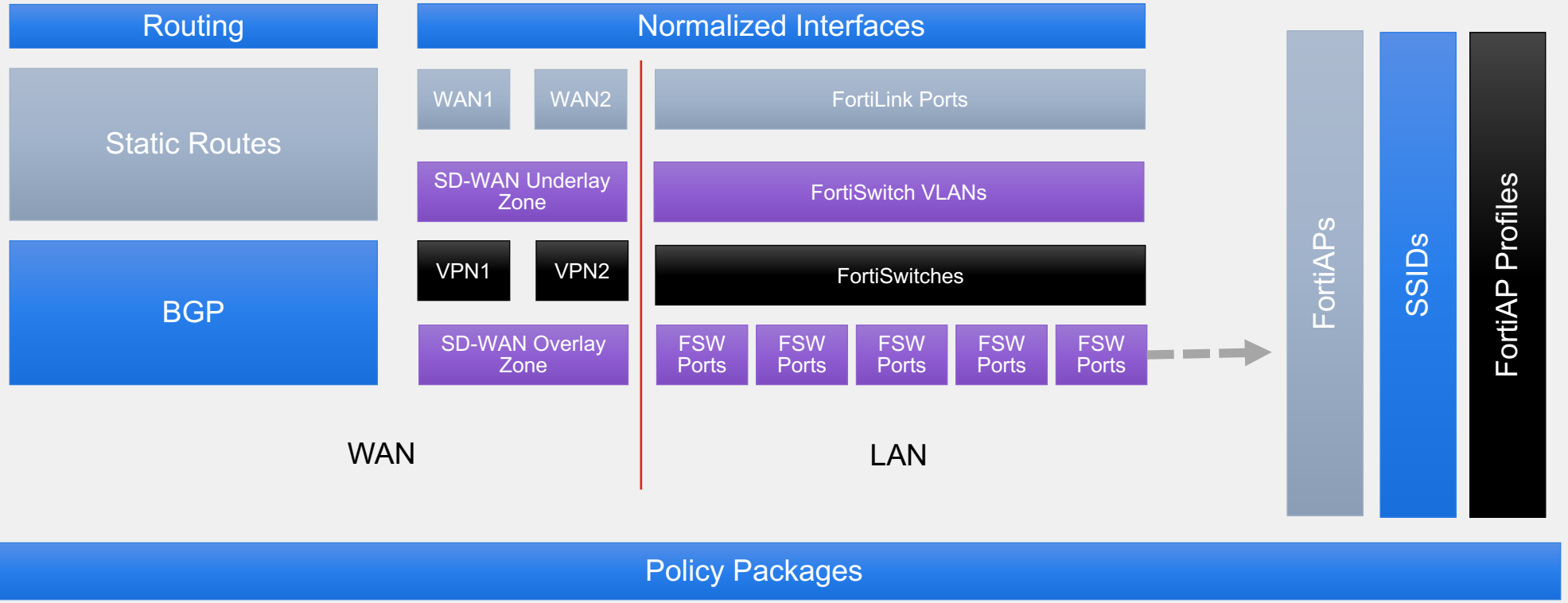
Inter-Layer Connections



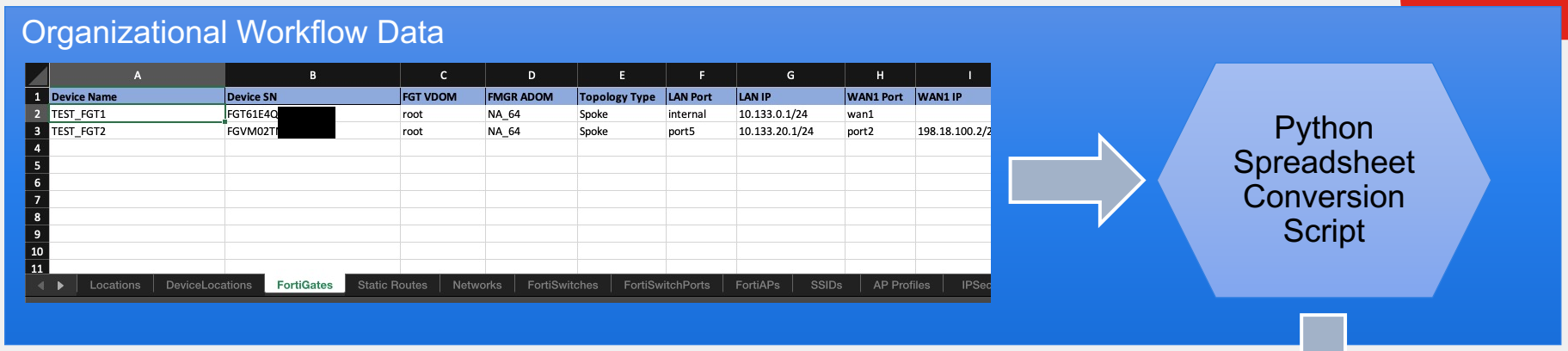
Included Example Spreadsheet

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Upstream FortiGate Name	FortiSwitch Name	Port	Description	Native VLAN	Allowed VLANs	Security Policy	LLDP Profile	QoS Policy	STP	Loop Guard	Edge Port	STP BPDU Guard	STP Root Guard	DHCP Snooping
2	TEST_FGT1	IDF_108E_1	1	User	BRANCH_CORE		DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
3	TEST_FGT1	IDF_108E_1	2	User	WIRELESS_VLAN20		DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
4	TEST_FGT1	IDF_108E_1	3	User	WIRELESS_VLAN20		DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
5	TEST_FGT1	IDF_108E_1	4	User	WIRELESS_VLAN20		DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
6	TEST_FGT1	IDF_108E_1	5	AP	IPT_VLAN30		DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
7	TEST_FGT1	IDF_108E_1	6	Printer	WIFI_MGMT_VLAN99		DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
8	TEST_FGT1	IDF_108E_1	7	User	BRANCH_CORE		DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
9	TEST_FGT1	IDF_108E_1	8	Trunk		all	DOT1X_TEST	default-auto-isl	default	enabled	disabled	enabled	disabled	disabled	untrusted
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															

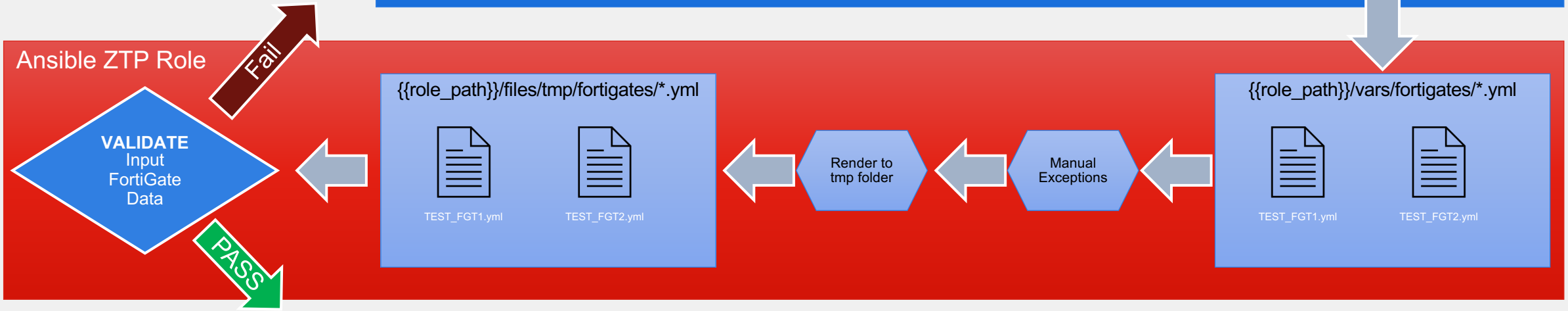
Locations | DeviceLocations | FortiGates | Static Routes | Networks | FortiSwitches | FortiSwitchPorts | FortiAPs | SSIDs | AP Profiles | IPSec | SDWAN Zones | SDWAN Interfaces | SDWAN Rules | SDWAN SLAs | BGP N

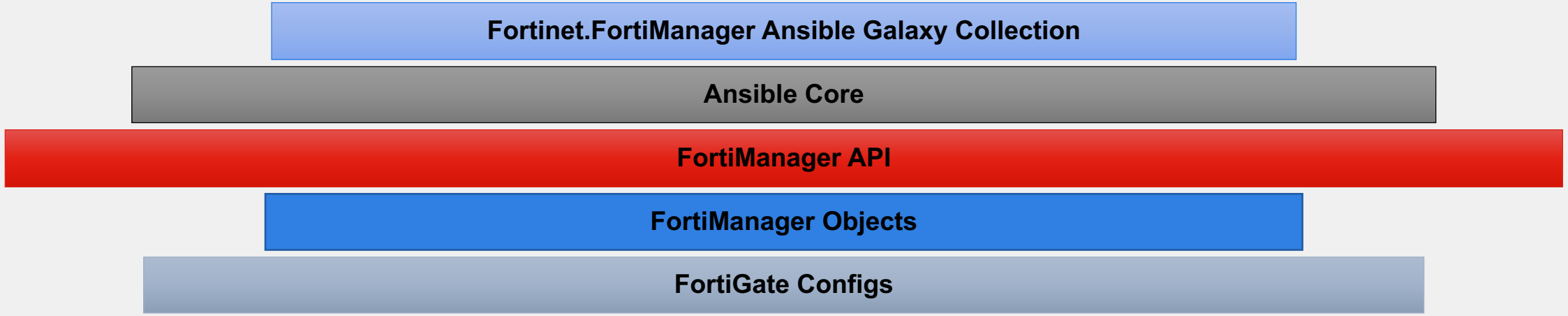


Included Spreadsheet Import

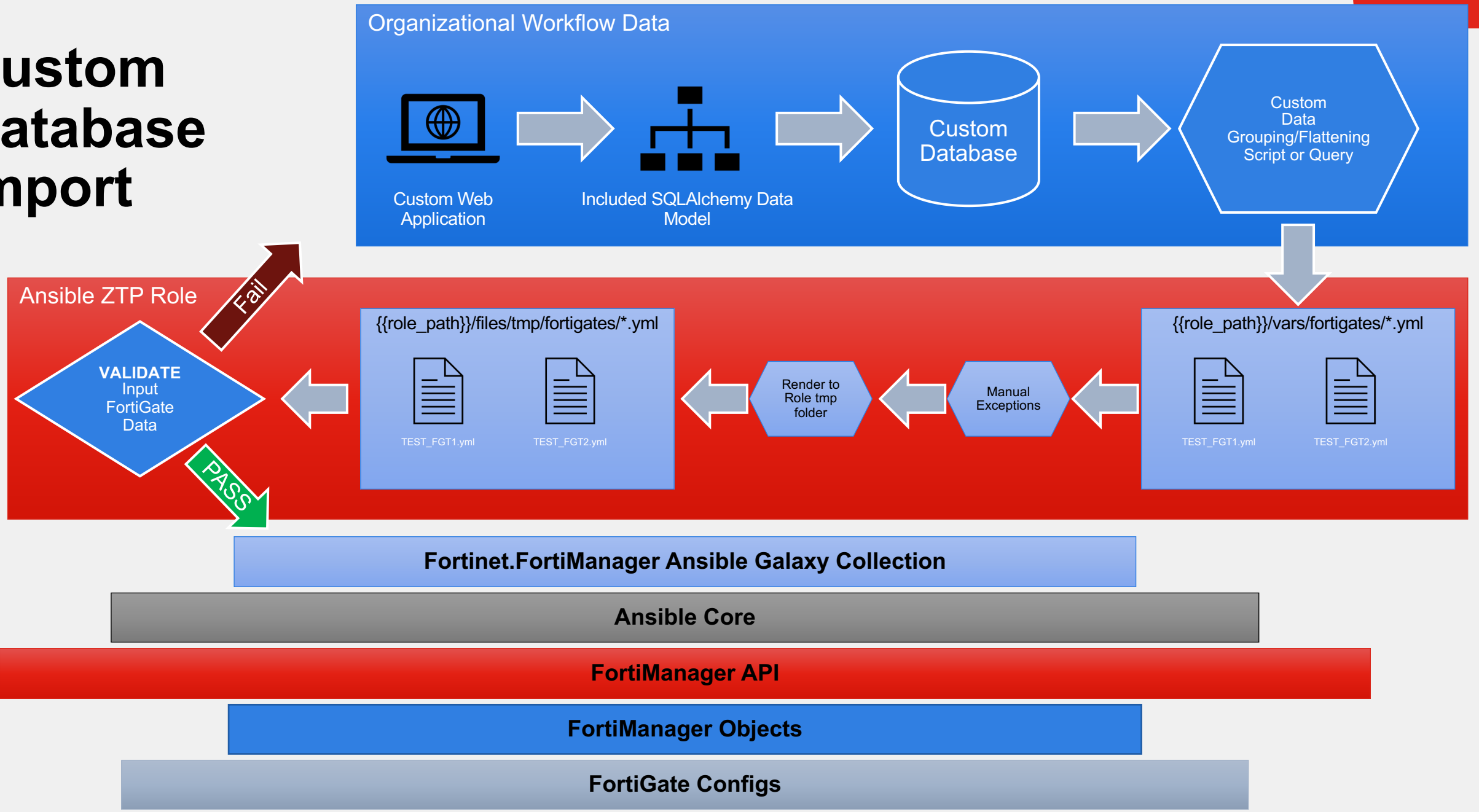


	A	B	C	D	E	F	G	H	I
1	Device Name	Device SN	FGT VDOM	FMGR ADOM	Topology Type	LAN Port	LAN IP	WAN1 Port	WAN1 IP
2	TEST_FGT1	FGT61E4Q[redacted]	root	NA_64	Spoke	internal	10.133.0.1/24	wan1	
3	TEST_FGT2	FGVM02T[redacted]	root	NA_64	Spoke	port5	10.133.20.1/24	port2	198.18.100.2/2
4									
5									
6									
7									
8									
9									
10									
11									



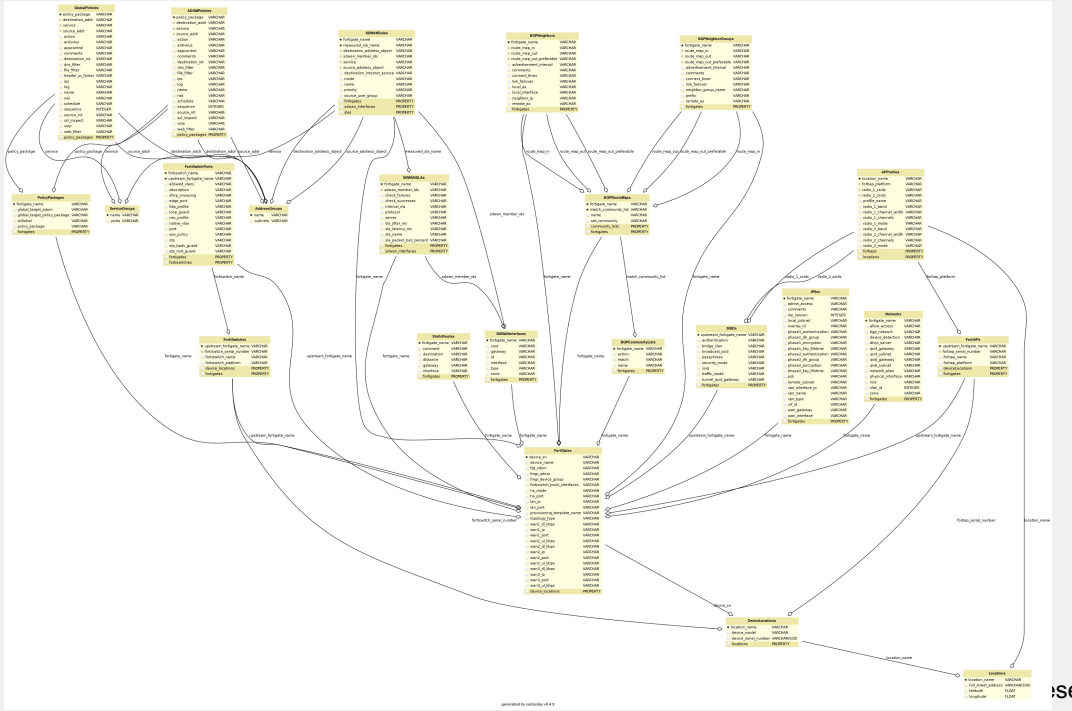


Custom Database Import



Organizational Workflow Data Model

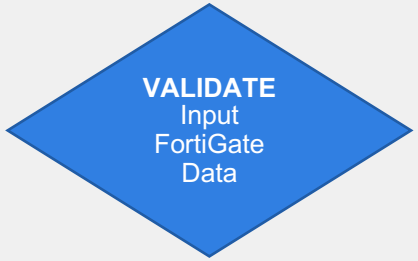
- We are using a spreadsheet as an included data source for importing FortiGate ZTP Data.
- However, spreadsheets are not for everyone.
- To aid in the design of an alternative input method, all spreadsheet tables (sheets), and attributes (columns) are modeled in Python SQLAchemy classes.
- This allows for the quick generation of a UML Diagram and custom applications.
- This is the *real value* of this Ansible Role – we've done **the data modeling** that organizations can use to accelerate their understanding of FortiGate ZTP.



```python
class Locations(Base):
    """Location Data"""
    __tablename__ = "locations"
    location_name = Column(String, primary_key=True)
    full_street_address = Column(VARCHAR(500), nullable=False)
    latitude = Column(Float, nullable=False)
    longitude = Column(Float, nullable=False)


class DeviceLocations(Base):
    """Device Location Data"""
    __tablename__ = "deviceLocations"
    __table_args__ = (
        PrimaryKeyConstraint('location_name'),
    )
    location_name = Column(String, ForeignKey("locations.location_name"))
    device_serial_number = Column(VARCHAR(500), nullable=False)
    device_model = Column(String, nullable=False)

    # Relationships
    locations = relationship("Locations")

class FortiGates(Base):
    """ FortiGate Data """
    __tablename__ = "fortigates"
    __table_args__ = (
        PrimaryKeyConstraint('device_sn'),
    )
    device_name = Column(String)
    device_sn = Column(String, ForeignKey("deviceLocations.device_serial_number"))
    fgt_vdom = Column(String)
    fmgr_adom = Column(String)
    topology_type = Column(String)
    lan_port = Column(String)
    lan_ip = Column(String)
    wan1_port = Column(String)
```

FortiGate YAML File Schema

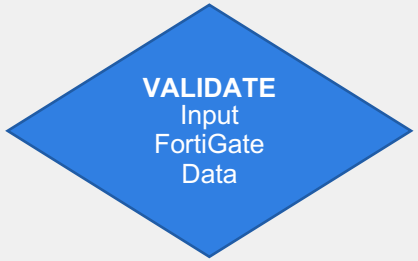
- The FortiGate YAML files must adhere to the “Ansible ZTP Role Data Model”.
- We have defined this data model via Python Schema and implemented an automatic check for all imported data.
- This schema can be extended by adding columns or sheets to the input spreadsheet (or custom database query).
- This is the data schema that the included Jinja2 templates have all been coded to use.
- If this schema changes, so must the Jinja2 templates.
- Ansible Tasks in the ZTP Role also depend on this data model.
- When Ansible ZTP Role runs, it reads every file as its own “dictionary” and merges all the files into a “list”.
- That “list of dictionaries” is then looped to run ZTP tasks, within the role.




```
- address_groups: <6 items>
  bgp_community_lists:
  -   action: permit
      fortigate_name: TEST_FGT1
      match: 65000:1
      name: HUB_VPN1
  - <4 keys>
  bgp_neighbor_groups:
  -   advertisement_interval: 2
      comments: Hub VPN BGP
      connect_timer: 10
      fortigate_name: TEST_FGT1
      link_failover: enable
      neighbor_group_name: HUB-VPN1
      prefix: 172.16.10.0/24
      remote_as: 65000
      route_map_in: HUB_VPN1
      route_map_out: null
      route_map_out_preferable: null
  - <11 keys>
  bgp_neighbors: <2 items>
  bgp_route_maps: <2 items>
  fap_profiles: <1 item>
  fap_ssids: <5 items>
  fgt_address_object_prefix: NA_BRANCH
  fgt_admin_user: admin
```

Validating Input Data

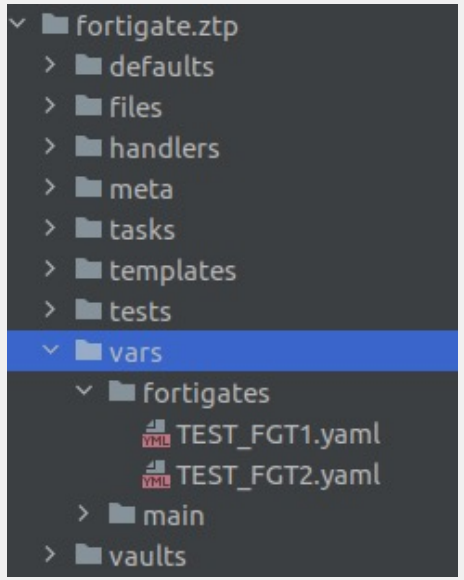
- We can validate all rendered FortiGate data to the “Ansible ZTP Role Data Model”.
- We’re using the Python “schema” package to accomplish this: https://pypi.org/project/schema/
- We have included Ansible playbooks to do the validation immediately after rendering the tmp FortiGate files.
- The role task can be found at: **tasks/templates/validate_rendered_fortigates.yml**
- The Python script this task calls can be found at: **files/python/ansible_ztp_role_data_model/validate_rendered_fortigates.py**
- The actual schema definition is a separate Python file that the script imports: **files/python/ansible_ztp_role_data_model/fortigate_schema.py**
- ^ *This file can be heavily modified to accommodate organization-specific workflows. It is shown on the right side of this slide.*
- This is the *real value* of this Ansible Role – we’ve done **the data validation** “legwork” that organizations can use to accelerate their own solutions.

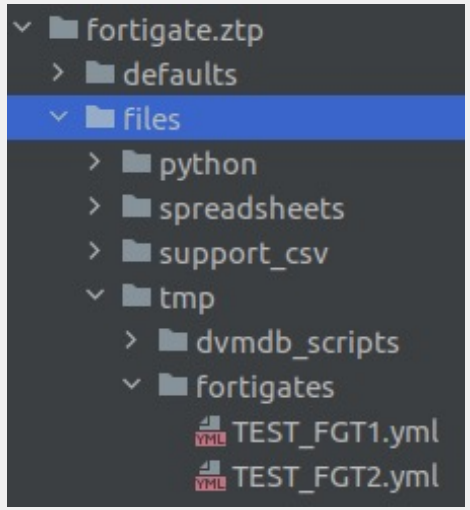


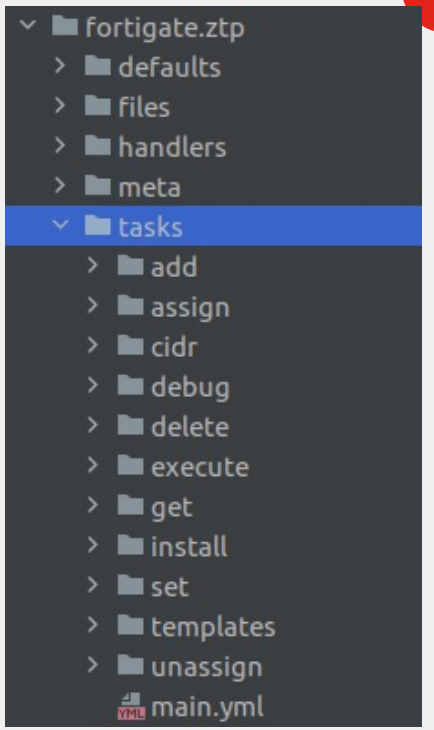

```
[{
    'address_groups': [
        {
            'name': And(str),
            'subnets': Or(str, list)
        }
    ],
    'bgp_community_lists': [
        {
            'action': And(str),
            'fortigate_name': And(str),
            'match': And(str),
            'name': And(str),
        }
    ],
    'bgp_neighbor_groups': [
        {
            'advertisement_interval': And(int),
            'comments': And(str),
            'connect_timer': And(int),
            'fortigate_name': And(str),
            'link_failover': And(str, Or('enable', 'disable')),
            'neighbor_group_name': And(str),
            'prefix': And(str),
            'remote_as': And(int),
            'route_map_in': Or(str, None),
            'route_map_out': Or(str, None),
            'route_map_out_preferable': Or(str, None),
        }
    ],
    'bgp_neighbors': [
        {
            'advertisement_interval': And(int),
            'comments': And(str),
            'connect_timer': And(int),
            'fortigate_name': And(str),
            'link_failover': And(str, Or('enable', 'disable')),
            'local_as': And(int),
            'local_interface': And(str),
```

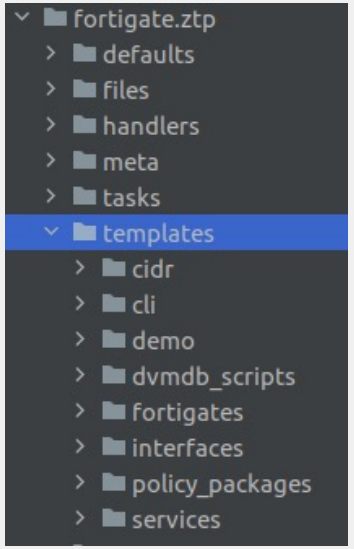
Ansible ZTP Role Folder Structure

- **/defaults** allow for “filling in the gaps” with configuration input.
- **/vars/fortigates** is where active configuration parameters live and is where imported data is sent.
- **/templates** contains Jinja2 code for the Ansible ZTP data layer.
- **/tasks** contains pre-configured API calls for creating ZTP objects.
- **/files/** contains python scripts, tmp files, spreadsheet inputs, and more.





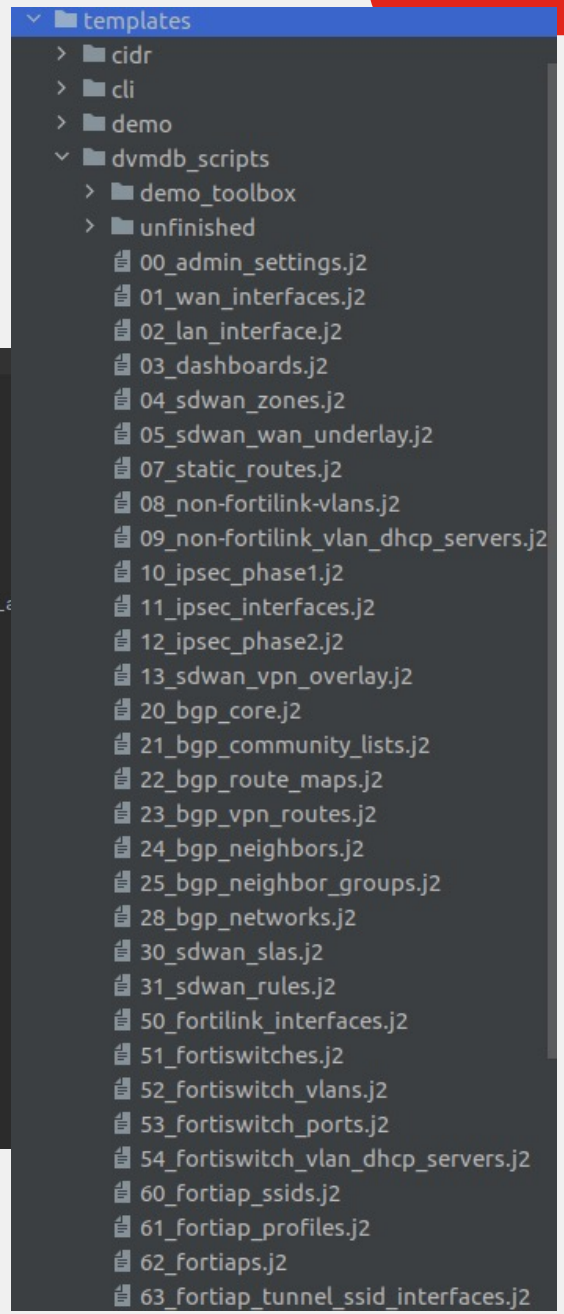




FortiGate Config Script Jinja2 Templates

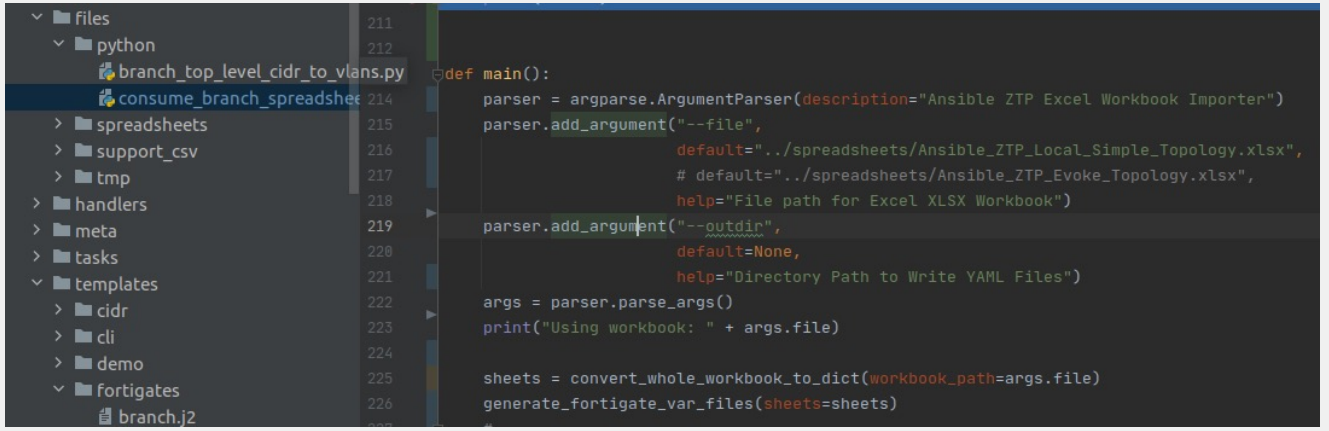
- Scripts have an intrinsic sort-order with naming schema.
- Each **templates/dvmdb_scripts/*.j2** script is run for every FortiGate YAML file under **files/tmp/fortigate/**
- Allows for full customization for FortiGate configurations.
- Can ”debug” script creation and simply copy the rendered scripts to another file location.
- The README.md documentation covers this topic in more detail.

```
{% if item.1.ipsec | length > 0 %}
config vpn ipsec phase1-interface
{% for vpn in item.1.ipsec %}
    edit "{{ vpn.vpn_name }}"
        set interface {{ vpn.wan_interface }}
        set ike-version {{ vpn.ike_version }}
        set net-device enable
        set mode-cfg enable
        set proposal {{ vpn.phase1_encryption | lower }}-{{ vpn.phase1_a
        set add-route disable
        set localid {{ vpn.fortigate_name }}
        set peertype any
        set idle-timeout enable
        set auto-discovery-receiver enable
        set network-overlay enable
        set network-id {{ vpn.overlay_id }}
        set remote-gw {{ vpn.wan_gateway }}
        set psksecret {{ vpn.psk }}
        set dpd-retrycount 2
        set dpd-retryinterval 2
        set keylife {{ vpn.phase1_key_lifetime }}
        set dhgrp {{ vpn.phase1_dh_group }}
    next
{% endfor %}
end
{% endif %}
```

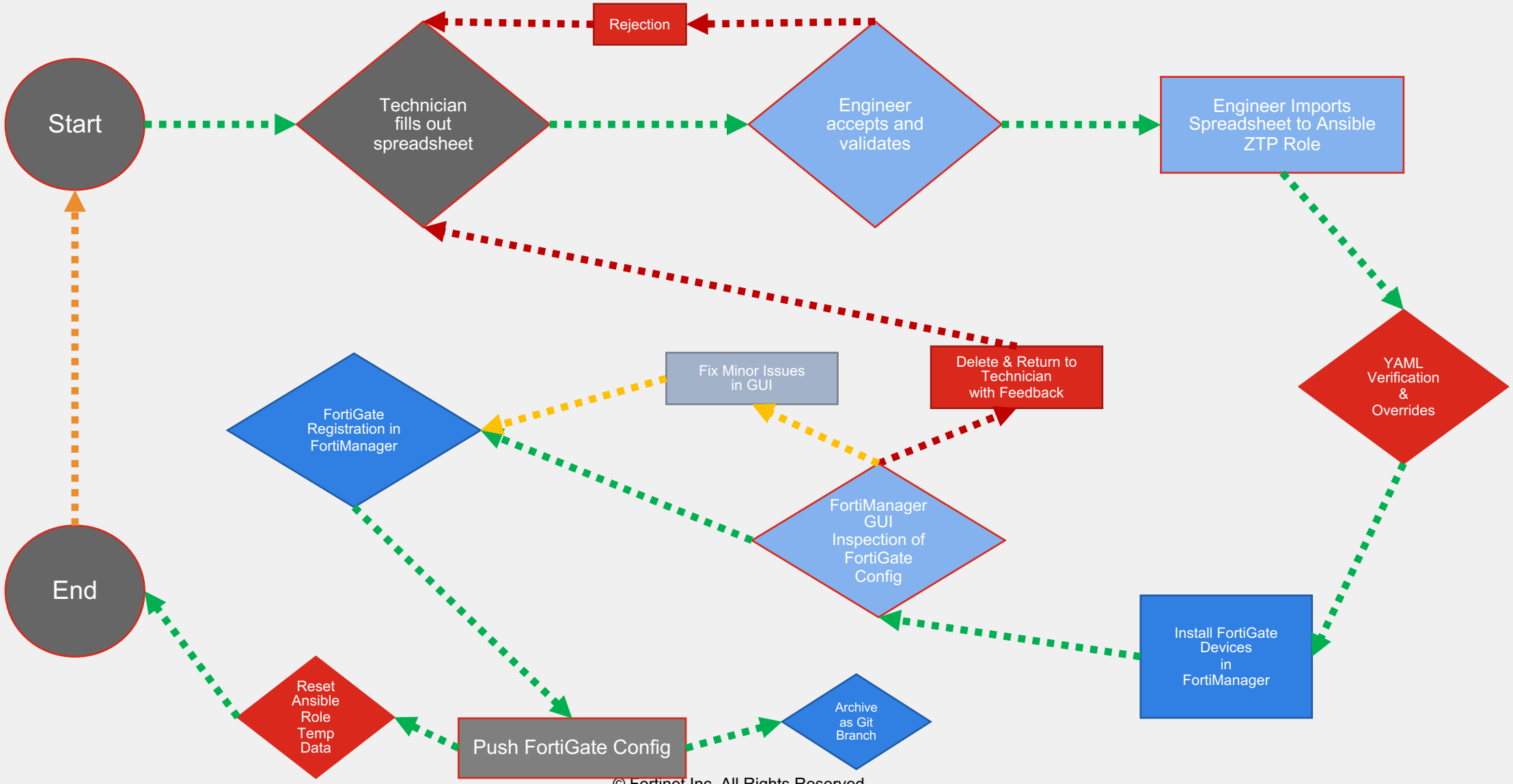


Spreadsheet to Ansible ZTP Data Conversion

- openpyxl Python library provides parsing.
- Follows the “Ansible ZTP Role Data Model”
- Ansible Role then processes imported data.
- Ansible Role can then be reset to accept a new deployment spreadsheet.



Deployment Workflow




FORTINET